INSTITUTO TECNOLÓGICO DE COSTA RICA
VICERRECTORÍA DE INVESTIGACIÓN Y EXTENSIÓN

INFORME FINAL DE PROYECTO DE INVESTIGACIÓN

ESCUELA DE MATEMÁTICA

Cartago, Costa Rica

2014

Tabla de contenido

Generalidades 5

Resumen y palabras claves 6

Lista de tablas 7

Lista de figuras 9

1. Apartado 8: Introducción **11**

1.1. Clasificación 11

1.2. Formulación del problema 12

1.2.1. Planteamiento formal del problema 18

1.3. Heurísticas de optimización 19

1.3.1. Uso de fórmulas de actualización 21

1.3.2. Algoritmos genéticos 23

1.3.3. Enjambres de partículas 38

1.3.4. Búsqueda tabú 46

1.4. Estudios previos en particionamiento 53

1.5. Objetivos del proyecto 57

1.6. Recursos empleados en el proyecto 58

2. Apartado 9: Metodología **61**

2.1. Algoritmo genético aplicado a particionamiento 61

2.1.1. Descripción del algoritmo genético 65

2.2. Algoritmo basado en enjambres de partículas en particionamiento 75

2.2.1. Descripción del algoritmo basado en enjambres de partículas 78

2.3. Búsqueda tabú aplicado a particionamiento 82

2.3.1. Preliminares 82
2.3.2. Generalidades de la implementación de los algoritmos de BT 83
2.3.3. Búsqueda tabú por transferencias 88
2.3.4. Búsqueda tabú mediante movimiento de centros de gravedad 90
2.4. Exploración de la variabilidad: optimización de parámetros 93
2.4.1. Algoritmos genéticos 95
2.4.2. Enjambres de partículas 100
2.4.3. Búsqueda tabú por transferencias 109
2.4.4. Búsqueda tabú mediante movimiento de centros de gravedad 111

3. Apartado 10: Resultados 113
3.1. Resultados en tablas de referencia 114
3.1.1. Resultados en algoritmos genéticos y enjambres de partículas 115
3.1.2. Resultados en búsqueda tabú 116
3.1.3. Resumen del rendimiento en las tablas de referencia 118
3.2. Resultados en tablas adicionales 121
3.3. Resultados en tablas clásicas de la literatura 124
3.4. Jerarquización de las heurísticas 127
3.5. Diseño de la heurística híbrida 128
3.5.1. Descripción de la heurística híbrida 129
3.5.2. Aplicación de la heurística híbrida 132
3.5.3. Análisis del rendimiento 135

4. Apartado 11: Discusión y conclusiones 139
4.1. Alcance de los objetivos 139
4.2. Otras conclusiones 141

5. Apartado 12: Recomendaciones 145

Apartado 14: Referencias 146

3. Título

HEURÍSTICAS DE OPTIMIZACIÓN COMBINATORIA PARA LA CLASIFICACIÓN DE DATOS. Código 5402-1440-3901

4. Autores y direcciones

M.Sc. Juan José Fallas Monge (jfallas@itcr.ac.cr)

M.Sc. Jeffry Chavarría Molina (jchavarria@itcr.ac.cr)

5. Participantes del proyecto

M.Sc. Juan José Fallas Monge (Coordinador)

M.Sc. Jeffry Chavarría Molina

6. Resumen

En el presente proyecto de investigación se realizó la implementación de las heurísticas de algoritmo genético, enjambre de partículas (EP) y búsqueda tabú (BT) para el estudio del problema de clasificación por particiones con datos cuantitativos. En total se implementaron cuatro algoritmos, dado que en el caso de búsqueda tabú se diseñaron dos variantes. Una de ellas constituyó en la generación de vecindarios mediante transferencias de individuos de una clase a otra; mientras que la otra consistió en la construcción de los vecinos mediante el movimiento de los centros de gravedad de las clases (BTCG).

Los algoritmos fueron aplicados a veinte tablas de datos tomadas de la literatura. Además, se diseñaron ocho tablas adicionales de mayor tamaño y complejidad, para verificar el rendimiento de los algoritmos. Se realizó, además, un análisis de la variabilidad de los resultados, en función de los parámetros de las diferentes heurísticas. Esto permitió determinar, para cada una de ellas, la combinación de parámetros que generó el mejor rendimiento posible en cada heurística. Además, se realizó una comparación de la eficiencia de las heurísticas implementadas, lo cual permitió generar una jerarquización de ellas como función del rendimiento, siendo BTCG y EP las que mostraron mejores resultados.

7. Palabras claves

Palabras claves: heurísticas, optimización, particionamiento de datos, análisis de datos, algoritmos.

Lista de tablas

1.1. Tabla de datos de tamaño $n \times p$. 18

1.2. Instancias y fitness asignados en la primera iteración. 32

1.3. Instancias y fitness asignados en la primera iteración. 32

1.4. Instancias y fitness asignados en la primera iteración. 32

1.5. Instancias y fitness asignados en la primera iteración. 33

2.1. Representación de código genético de una instancia. 63

2.2. Representación de código genético de una instancia para $n = 10$ y $k = 4$. . 63

2.3. Representación de los individuos a clasificar. 64

2.4. Tabla de inercia intra-clase y valor de aptitud para cada instancia. 71

2.5. Porcentajes de atracción y tiempos promedios de ejecución en segundos para diferentes umbrales en la convergencia de $k-$medias. 74

2.6. Comparación entre el uso del líder actual y el líder global en la fórmula de actualización de la velocidad. 80

2.7. Porcentajes de atracción y tiempos promedios de ejecución en segundos para diferentes umbrales en la convergencia de $k-$medias. 81

2.8. Resultados obtenidos en la heurística de BT por transferencias, en pruebas preliminares realizadas para analizar si se aplicaba o no un proceso de intensificación. 85

2.9. Resultados obtenidos en la heurística de BT mediante movimiento de centros de gravedad, en pruebas preliminares realizadas para analizar si se aplicaba o no un proceso de intensificación. 85

2.10. Resumen de mejores parámetros para enjambre de partículas. 108

3.1. Codificación de las tablas de referencia. 114

3.2. Resultados generados por algoritmos genéticos (AG) en las 16 tablas de referencia, en 500 inicializaciones al azar. 116

3.3. Resultados en enjambres de partículas (OEP) para las 16 tablas de referencia, en 500 inicializaciones al azar. 117

3.4. Resultados obtenidos en las 16 tablas de referencia en los algoritmos de búsqueda tabú, en 500 inicializaciones al azar. 118

3.5. Rendimiento de los algoritmos en las tablas de referencia. 119

3.6. Codificación de las tablas adicionales de 1050 y 2100 individuos. 121

3.7. Resultados obtenidos en las tablas A17, . . . , A24 en el algortimo genético, en 500 inicializaciones al azar. 122

3.8. Resultados obtenidos en las tablas A17, . . . , A24 en los algoritmos de enjambres de partículas, en 500 inicializaciones al azar. 122

3.9. Resultados obtenidos en las tablas A17, . . . , A24 en los algoritmos de búsqueda tabú, en 500 inicializaciones al azar. 123

3.10. Resumen del rendimiento de los algoritmos en las tablas A17, . . . , A24. . . 123

3.11. Rendimiento de los algoritmos en tablas clásicas de la literatura. 126

3.12. Parámetros en la prueba de la heurística híbrida. 134

3.13. Resultados obtenidos en los casos en análisis. 135

3.14. Resultados obtenidos con EP, aplicando $k-$medias, desactivando búsqueda tabú e incrementando el número de partículas en el enjambre. 137

4.1. Combinación de parámetros generados en el algoritmo de enjambre de partículas. 143

Lista de figuras

1.1. Representación cromosómica de una solución con tres genes, cada uno con una codificación binaria de diferente tamaños. 28

1.2. Representación de un cromosoma con cuatro genes de diferente longitud y con codificación diferente en cada uno de los genes. 29

1.3. Ejemplo de una ruleta simple para 6 instancias I_1, I_2, I_3, I_4, I_5 e I_6 con fitness respectivo 2, 6, 4, 6, 5 y 4. 31

1.4. Cruce basado en un punto de corte. 34

1.5. Cruce basado en dos puntos de corte. 35

2.1. Construcción de la ruleta para el método SUS. 72

2.2. Lista tabú en transferencias. 89

2.3. Superficie generada por el porcentaje de atracción en la tabla de datos Tabla-105-K(7)Card(≠)DS(≠). 97

2.4. Superficie generada por el porcentaje de atracción en la tabla de datos Tabla-525-K(7)Card(≠)DS(≠). 99

2.6. Mapa de contornos generado por el porcentaje de atracción para un α fijo y una malla regular bidimensional de 0.25 de grosor, producto de la variación de los parámetros w y c_1. 102

2.5. Mapa de contornos generado por el porcentaje de atracción para un α fijo y una malla regular bidimensional producto de la variación de los parámetros w y c_1. 103

2.7. Mapa de contornos generado por el porcentaje de atracción para un α fijo y una malla regular bidimensional producto de la variación de los parámetros w y c_1. 106

2.8. Curvas generadas por el porcentaje de atracción producto de la variación del parámetro $\alpha \in [-2, 2]$ en una malla unidimensional de grosor 0.125, con $w = 1.75$ y $c_1 = 1.25$. 107

2.9. Curvas generadas por el porcentaje de atracción producto de la variación del parámetro $\alpha \in [-2, 2]$ en una malla unidimensional de grosor 0.125, con $w = 4.25$ y $c_1 = 3.75$. 108

2.10. Panel de control del algoritmo de BT por transferencias. 109

2.11. Resultados obtenidos al variar el parámetro ReiniSolActualCada en la heurística BT por transferencias, en la tabla de 105 individuos. 110

2.12. Resultados obtenidos al variar el parámetro ReiniSolActualCada en la heurística BT por transferencias, en la tabla de 525 individuos. 111

2.13. Resultados obtenidos al variar el parámetro ReiniSolActualCada en la heurística BT mediante movimiento de CG, en la tabla de 105 individuos. 112

2.14. Resultados obtenidos al variar el parámetro ReiniSolActualCada en la heurística BT mediante movimiento de CG, en la tabla de 525 individuos. 112

Capítulo 1

Apartado 8: Introducción

En el presente capítulo se realiza una revisión general del tema de clasificación por particiones, así como los principales conceptos y resultados en este tema. Además, se detalla sobre las heurísticas de optimización combinatoria que fueron aplicadas en el proyecto.

1.1. Clasificación

Los métodos de agrupamiento o de clasificación consisten en una serie de estrategias que buscan determinar grupos (denominados *clusters* en la literatura en inglés), bajo el principio generalizado de que los objetos (individuos) pertenecientes a un mismo grupo, presenten características de mayor similitud entre sí (con respecto a algún criterio seleccionado previamente), comparado con los individuos que quedaron asignados a otros grupos. Según Berzal (2005), las técnicas de agrupamiento analizan el conjunto de observaciones disponibles y permiten realizar una clasificación asignando cada observación a un grupo, de forma que cada uno de ellos sea más o menos homogéneo y diferenciable de los demás.

En términos generales, el problema de clasificación parte de un conjunto de n individuos y analiza la forma de distribuirlos en K grupos (clases). Si se aborda exhaustivamente,

dicha distribución generaría un problema de dificultad exponencial, dado que acorde con la fórmula de Stirling que proporciona el número Φ de posibles formas de cómo distribuir n individuos en K clases, se tiene que (Trejos et al., 2014):

$$\Phi = \frac{1}{K!}\sum_{i=0}^{K}\left((-1)^{K-i}\binom{K}{i}i^n\right).$$

Por ejemplo, en un problema de 7 clases y 105 individuos se obtiene

$$\Phi = \frac{1}{7!}\sum_{i=0}^{7}\left((-1)^{7-i}\binom{7}{i}i^{105}\right) \approx 1.0786 \times 10^{85}.$$

Por lo tanto, es evidente la inviabilidad de explorar todas las posibles formas de realizar la distribución de los objetos en los diferentes grupos. Además, esto justifica a su vez la necesidad de contar con algoritmos o estrategias complementarias que permitan proporcionar una solución satisfactoria, en un tiempo razonable.

1.2. Formulación del problema

El problema de particionamiento corresponde a un problema de optimización combinatoria en el que se desea efectuar una distribución de individuos en grupos, regido por algún criterio mínimo de costo, el cual se fundamenta en el grado de similitud que presenten los individuos asignados a un mismo grupo. Para efecto de precisar dicho problema, a continuación se presentan algunos conceptos necesarios para la formulación.

Primero se considera el conjunto de individuos $X = \{x_1, \ldots, x_n\}$ y se quiere construir un agrupamiento de dichos objetos en K grupos. Dicho agrupamiento se denomina una partición del conjunto X y se define formalmente en la definición 1.

DEFINICIÓN **1** *Una partición P del conjunto $X = \{x_1, \ldots, x_n\}$ en K clases es un conjunto $P = \{C_1, \ldots, C_K\}$ tal que :*

- $C_l \subset X$ *y* $C_l \neq \emptyset$ *para todo* l.
- $l \neq l' \Rightarrow C_l \cap C_{l'} = \emptyset$.
- $X = \bigcup_{l=1}^{K} C_l$.

Por otra parte, se tiene el concepto de *peso* de un individuo, que representa el grado de importancia o representatividad que tiene un individuo específico con respecto a los demás. Si p_i denota el peso del individuo x_i de X, entonces debe satisfacerse que $0 < p_i < 1$, para $i = 1, \ldots, n$. Además,

$$\sum_{i=0}^{n} p_i = 1.$$

De manera similar, se considera el concepto de *peso de una clase*, el cual indica la importancia medida en términos porcentuales, de una clase con respecto a las demás. De manera muy razonable, se define el peso μ_l de la clase C_l, para $l = 1, \ldots, K$, como la suma de los pesos de los individuos que la componen. Esto es,

$$\mu_l = \sum_{x_i \in C_l} p_i.$$

A partir de los conceptos de peso de un individuo y peso de una clase, se define el *centro de gravedad* de una clase. A continuación se establece dicho concepto, junto con la definición del centro de gravedad de X.

DEFINICIÓN **2** *El centro de gravedad de la clase C_l se define como:*

$$g_l = \frac{1}{\mu_l} \sum_{x_i \in C_l} p_i x_i.$$

Además, el centro de gravedad de X se define como:

$$g = \sum_{x_i \in X} p_i x_i.$$

Para efectos del proyecto, se asume que todos los individuos por clasificar tienen el mismo peso, por lo que se considera $p_i = \frac{1}{n}$, para $i = 1, \ldots, n$. Además, si $\mathsf{Card}(C_l)$ denota la cardinalidad de la clase C_l, entonces:

$$\mu_l = \sum_{x_i \in C_l} p_i = \sum_{x_i \in C_l} \frac{1}{n} = \frac{\mathsf{Card}(C_l)}{n}.$$

Bajo ese mismo supuesto, la definición de centro de gravedad de la clase C_l se expresa como:

$$g_l = \frac{1}{\mu_l} \sum_{x_i \in C_l} p_i x_i = \frac{n}{\mathsf{Card}(C_l)} \sum_{x_i \in C_l} \frac{1}{n} x_i = \frac{1}{\mathsf{Card}(C_l)} \sum_{x_i \in C_l} x_i.$$

Por lo tanto, el centro de gravedad así definido para una clase, se puede interpretar como un individuo promedio de los individuos que pertenecen a dicha clase. Además, asumiento nuevamente igualdad de los pesos, el centro de gravedad asociado a X queda expresado por:

$$g = \sum_{x_i \in X} \frac{1}{n} x_i = \frac{1}{n} \sum_{x_i \in X} x_i.$$

Por otra parte, en el agrupamiento con datos debe definirse la medida de similitud que se utilizará. Dicha medida se conoce como una métrica y corresponde a una función de distancia que permite cuantificar qué tan similares, o disímiles, son una pareja de individuos entre sí. Dependiendo del problema a estudiar, puede que una métrica sea más adecuada que otra. Entre las métricas más comunes están la distancia euclídea normalizada, la distancia euclídea ponderada y la distancia de Mahalanobis (Berzal, 2005). En general, la distancia euclídea corresponde a la forma clásica de medir la distancia en problemas de agrupamiento con datos cuantitativos. En este sentido, si x_i y x_j son vectores de $\mathbb{R}^p$,

entonces la distancia euclídea d entre estos dos vectores cumple con la relación:

$$d^2(x_i, x_j) = (x_i - x_j)^T (x_i - x_j) = \|x_i - x_j\|^2 .$$

En el proyecto se utilizó esta distancia para decidir sobre la similitud o disimilitud entre dos objetos. La interpretación se realizó en el sentido que un valor alto de $\|x_i - x_j\|^2$, indica una alta disimilitud entre los individuos x_i y x_j, teniendo una interpretación recíproca para el caso de valores pequeños en la distancia.

A partir de la función de distancia seleccionada, es necesario precisar el criterio que se quiere optimizar en el problema de particionamiento. En este sentido, a continuación se define el concepto de inercia intraclases asociada a una partición P del conjunto X.

DEFINICIÓN **3** *Sea $P = \{C_1, \ldots, C_K\}$ una partición de X. Si $g_1, \ldots, g_K$ son los centros de gravedad de las clases y $p_i = \frac{1}{n}$ para $i = 1, \ldots, n$, entonces se llama inercia intraclases de P a:*

$$W(P) = \frac{1}{n} \sum_{l=1}^{K} \sum_{x_i \in C_l} \|x_i - g_l\|^2 .$$

Así definida, la inercia intraclases $W(P)$ permite cuantificar el agrupamiento de los individuos en todas las clases a la vez. Cuanto menor sea el valor de $W(P)$, los individuos pertenecientes a una misma clase están más agrupados entre sí, indicando por ende una mayor similitud a lo interno de las clases. Lo anterior en contraposición de la disimilitud que presentan los individuos pertenecientes a clases diferentes.

Por otra parte, de una manera complementaria se puede definir la inercia interclases asociada a una partición P del conjunto X, la cual, bajo las mismas hipótesis enunciadas

en la definición 3, está dada por

$$B(P) = \sum_{l=1}^{K} \mu_l \|g_l - g\|^2 = \sum_{l=1}^{K} \frac{\mathsf{Card}(C_l)}{n} \|g_l - g\|^2 .$$

La inercia interclases $B(P)$ permite cuantificar la separación de las clases entre sí. Entre más alto sea su valor, entonces existe mayor diferencia entre las clases generadas por la partición P.

En el problema de particionamiento abordado se busca lograr los objetivos complementarios de minimizar el valor de $W(P)$ y maximizar el valor de $B(P)$, permitiendo esto que los individuos asignados a una misma clase sean los más similares entre sí, aunado de una asignación de individuos muy disímiles a clases diferentes. Por lo tanto, el problema de particionamiento de datos puede ser abordado como un problema de máxima inercia interclases o de mínima inercia intraclases. Ambos enfoques tienen un caracter equivalente, tal y como lo permite concluir el teorema de Fisher, cuya prueba puede ser consultada en Trejos et al. (2014), y que se cita a continuación.

TEOREMA **1** *Dada una partición $P = \{C_1, \ldots, C_K\}$ de X, g el centro de gravedad de X y si $I(X)$ denota la inercia total definida por*

$$I(X) = \sum_{i=1}^{n} p_i \|x_i - g\|^2 ,$$

entonces $I(X) = B(P) + W(P)$.

Dado que la inercia total $I(X)$ corresponde a la suma de las n distancias al cuadrado ponderadas[1] de los individuos x_i, para $i = 1, \ldots, n$, a g que es el centro de gravedad de X

[1]La norma cuadrada $\|x_i - g\|^2$, que representa la distancia al cuadrado entre el individuo x_i y g que es el centro de gravedad de X, se pondera con el peso p_i relativo al individuo x_i.

(ver definición 2), entonces $I(X)$ es constante. Como consecuencia, del teorema de Fisher se concluye que la inercia interclases es máxima cuando la inercia intraclases es mínima y viceversa. Por lo tanto, el problema de particionamiento puede ser estudiado, tal y como ya se ha dicho, como un problema de máximo o de mínimo. En particular, en el proyecto se siguió el enfoque de hacer mínima la inercia intraclases, tal y como se establece en el planteamiento formal del problema en la sección 1.2.1.

Finalmente, se hace énfasis en que en el proyecto el número K de clases se asume como fijo, esto es, se parte de la premisa que se quiere particionar los n individuos de X en K clases, siendo K un parámetro previamente seleccionado. Esta decisión, tal y como se indica en Trejos et al. (2014), se basa en que la inercia intraclases decrece de manera monótona cuando el número de clases aumenta. Esta proposición se formaliza en el teorema 2.

TEOREMA **2** *Si $K < n$, P_K^* denota la mejor partición de X en K clases y P_{K+1}^* denota la mejor partición de X en $K+1$ clases, entonces debe darse que:*

$$W(P_{K+1}^*) \leq W(P_K^*).$$

Por esta razón, si el número de clases es variable, entonces carece de sentido establecer el problema de hacer mínima a la inercia intraclases. En efecto, en ese caso bastaría considerar $K = n$ (esto es, construir todas las clases compuestas por un único individuo, en cuyo caso el centro de gravedad de cada clase sería el mismo individuo que la compone) y notar que la inercia intraclases asociada a la partición P_n tomaría el valor de cero. En virtud de lo anterior se fundamenta la decisión de tomar a K fijo en cada uno de los algoritmos.

1.2.1. Planteamiento formal del problema

A partir de las definiciones y premisas ya establecidas, a continuación se procede a concretizar los principales aspectos del problema de particionamiento de datos cuantitativos. Este problema parte de la tabla de datos de tamaño $n \times p$, caracterizada por n individuos y p variables cuantitativas independientes, que se representa en la tabla 1.1.

Tabla 1.1: Tabla de datos de tamaño $n \times p$.

Ind/Var	v_1	v_2	$\ldots$	v_p
x_1	x_{11}	x_{12}	$\ldots$	x_{1p}
x_2	x_{21}	x_{22}	$\ldots$	x_{2p}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_n	x_{n1}	x_{n2}	$\ldots$	x_{np}

La fila i de dicha tabla contiene las entradas del $i-$ésimo individuo que puede interpretarse como un vector en $\mathbb{R}^p$. Cada individuo se asumirá con peso constante $p = \frac{1}{n}$, para $i = 1, \ldots, n$, y x_{ij} corresponde al valor que toma x_i en la $j-$ésima variable cuantitativa v_j.

Si se considera el conjunto de n elementos $X = \{x_1, \ldots, x_n\}$ como representación matricial de la tabla 1.1, con $x_i \in \mathbb{R}^p$ y P una partición de X en K clases $C_1, \ldots, C_K$, entonces el problema de particionamiento de los individuos $x_1, \ldots, x_n$ en K clases, puede formularse como la minimización de la función:

$$W(P) = \frac{1}{n} \sum_{l=1}^{K} \sum_{x_i \in C_l} \|x_i - g_l\|^2 ,$$

donde $\|\cdot\|$ corresponde a la norma inducida por la métrica euclídea clásica, tal que:

$$\|x_i - g_l\|^2 = d^2(x_i, g_l) = (x_i - g_l)^T (x_i - g_l).$$

1.3. Heurísticas de optimización

El problema de optimización de particionamiento con datos cuantitativos, tal y como se ha planteado, busca minimizar la función que modela la inercia intraclases asociada a las diferentes posibles particiones P de X. Tal función está dada por:

$$W(P) = \frac{1}{n} \sum_{l=1}^{K} \sum_{x_i \in C_l} \|x_i - g_l\|^2 ,$$

y tiene la característica de no ser una función convexa, lo cual podría implicar la existencia de una diversidad de mínimos locales (Berzal, 2005; Ng & Wong, 2002; Sarkar & Yegnanarayana, 1996) . Esta característica de multiplicidad de óptimos relativos es lo que causa que los algoritmos tradicionales usados en particionamiento de datos, tal como el algoritmo de k-medias, encuentren mayoritariamente mínimos locales (Trejos & Murillo, 2004). En este sentido, en Babu & Murty (1994) se argumenta que dichos algoritmos convergen a un mínimo local, salvo que se conozca información adicional sobre el agrupamiento natural de los datos, lo cual normalmente no sucede.

En todo caso, los algoritmos de optimización global (tales como programación lineal, métodos de intervalos, ramificación y acotamiento, etc.) presentan una alta sensibilidad en tablas de datos de dimensión relativamente grande, en las cuales la probabilidad de que estos algoritmos detecten el agrupamiento óptimo, es prácticamente nula o al menos muy baja. En esos casos, dichos algoritmos reportan soluciones o agrupamientos que difieren significativamente del óptimo (Bagirov & Mardeneh, 2006). Acorde con Babu & Murty (1994), la mayoría de los métodos tradicionales usados en particionamiento de datos corresponden a técnicas de descenso de gradiente que, por su naturaleza de optimalidad local, convergen a soluciones suboptimales de la función objetivo. Estas características representan un fundamento para la necesidad de buscar estrategias de optimización alter-

nativas o complementarias, entre las cuales se encuentran las heurísticas de optimización combinatoria.

En los últimos años las heurísticas han sido usadas con el objetivo de resolver problemas complejos de optimización y es su naturaleza aleatoria lo que hace que evadan eficientemente la convergencia hacia óptimos locales (Babu & Murty, 1994). Según estos autores, entre las técnicas estocásticas más importantes están el sobrecalentamiento simulado (del inglés *simulated annealing*) y los algoritmos genéticos (del inglés *genetics algorithms*); teniendo estas dos heurísticas su fundamento en procesos industriales y naturales, respectivamente.

En el caso de las ciencias sociales y de la ingeniería, las técnicas de análisis de datos requieren de la solución de problemas complejos de optimización combinatoria. Sin embargo, tal y como ya se indicó, a pesar de que los algoritmos de búsqueda de soluciones óptimas pueden ser usados para resolver problemas pequeños, el tamaño inherente y la complejidad de los problemas que se presentan en la práctica, generan la necesidad de aplicar enfoques heurísticos (Brusco, 1999). Inclusive, hoy en día los algoritmos de agrupamiento más utilizados en problemas reales suelen basarse en heurísticas (Berzal, 2005).

No obstante, no debe perderse la perspectiva de que el uso de las heurísticas no debe extenderse a problemas en los que la complejidad no lo amerite. En esta misma línea, en Talbi (2009) se advierte que no es prudente aplicar heurísticas para resolver problemas para los cuales hay disponibilidad de algoritmos exactos que logran el cometido con eficiencia. Estos autores argumentan además que si los algoritmos exactos ofrecen una solución en un tiempo razonable, no tiene lógica recurrir a métodos probabilísticos como lo son las heurísticas. En cualquier caso, como se ha fundamentado, el problema de particionamiento de datos para tablas relativamente grandes, no puede ser resuelto por ningún algoritmo exacto en un tiempo razonable.

1.3.1. Uso de fórmulas de actualización

Una transferencia de un individuo x de la clase C_r a la clase C_l, genera una nueva partición P' de X, la cual difiere de la partición anterior P, entre otras cosas, por la variación sufrida por los centros de gravedad de las clases r y l, así como el cambio que se genera en la inercia intraclases. En términos computacionales, el cálculo directo de dichas variaciones obliga a ejecutar varios ciclos que podrían incidir de manera significativa en los tiempos de ejecución de los algoritmos.

Para solventar esta situación en los algoritmos basados en transferencias, se pueden emplear las denominadas fórmulas de actualización, las cuales permiten calcular los centros de gravedad de las nuevas clases $C'_r = C_r - \{x\}$ y $C'_l = C_l \cup \{x\}$, a partir de manipulaciones numéricas con un menor costo computacional. Además, dichas fórmulas permiten medir la variación generada en la inercia intraclases, como consecuencia de la transferencia del individuo x. En este sentido, en Trejos et al. (2014) se enuncian y se demuestran dichas fórmulas, las cuales se detallarán en los teoremas 3 y 4.

TEOREMA **3** *Al hacer la transferencia de x de la clase C_r a la clase C_l, los centros de gravedad se modifican de la siguiente manera:*

$$g(C_r - \{x\}) = \frac{1}{\mu_r - p_x}(\mu_r g_r - p_x x),$$

$$g(C_l \cup \{x\}) = \frac{1}{\mu_l + p_x}(\mu_l g_l + p_x x),$$

donde p_x es el peso del individuo x.

Como un caso particular, si los n individuos de X se asumen con el mismo peso $p = \frac{1}{n}$ y si $\mathsf{Card}(C_r)$ denota la cardinalidad de C_r, entonces $\mu_r = \frac{\mathsf{Card}(C_r)}{n}$. Por lo tanto, las fórmulas

de actualización de centros de gravedad pueden simplificarse tal y como se muestra a continuación:

$$\begin{aligned} g(C_r - \{x\}) &= \frac{1}{\mu_r - p_x}(\mu_r g_r - p_x x) \\ &= \frac{1}{\frac{\mathsf{Card}(C_r)}{n} - \frac{1}{n}}(\frac{\mathsf{Card}(C_r)}{n} g_r - \frac{1}{n} x) \\ &= \frac{1}{\mathsf{Card}(C_r) - 1}(\mathsf{Card}(C_r) g_r - x). \end{aligned}$$

De manera similar se obtiene que:

$$g(C_l \cup \{x\}) = \frac{1}{\mu_l + p_x}(\mu_l g_l + p_x x) = \frac{1}{\mathsf{Card}(C_l) + 1}(\mathsf{Card}(C_l) g_l + x).$$

Por su parte, en el teorema 4 se enuncia la forma en la que varía la inercia intraclases, como consecuencia de la misma transferencia.

TEOREMA **4** *Al transferir el individuo x de la clase C_r a la clase C_l, la inercia intraclases presenta la variación:*

$$\Delta W = \frac{\mu_r p_x}{\mu_r - p_x} ||g_j - x||^2 - \frac{\mu_l p_x}{\mu_l + p_x} ||g_l - x||^2.$$

Nuevamente, si se asumen pesos iguales, entonces se tiene que:

$$\frac{\mu_r p_x}{\mu_r - p_x} = \frac{\frac{\mathsf{Card}(C_r)}{n} \cdot \frac{1}{n}}{\frac{\mathsf{Card}(C_r)}{n} - \frac{1}{n}} = \frac{\mathsf{Card}(C_r)}{n(\mathsf{Card}(C_r) - 1)}.$$

De manera similar,

$$\frac{\mu_l p_x}{\mu_l + p_x} = \frac{\mathsf{Card}(C_l)}{n(\mathsf{Card}(C_l) + 1)}.$$

Por lo tanto, en este caso la variación de la inercia queda expresada como:

$$\Delta W = \frac{\mathsf{Card}(C_r)}{n(\mathsf{Card}(C_r)-1)}||g_r - x||^2 - \frac{\mathsf{Card}(C_l)}{n(\mathsf{Card}(C_l)+1)}||g_l - x||^2.$$

En la fórmula anterior se nota que para determinar ΔW no se necesita el cálculo de los centros de gravedad $g(C_r - \{x\})$ y $g(C_l \cup \{x\})$, por lo que dichos cálculos se ejecutan solo en caso de que la transferencia de x sea aceptada. Esto último permite disminuir la cantidad de operaciones realizadas en los algoritmos de transferencias y así se hizo en las implementaciones.

En las siguientes secciones se realiza una revisión detallada de los principales aspectos de las heurísticas implementadas.

1.3.2. Algoritmos genéticos

Los algoritmos genéticos son heurísticas basadas en el principio biológico de la evolución de las especies y fueron desarrollados por John Holland en la década de los años 70 en la Universidad de Michigan, Estados Unidos (de los Cobos et al., 2010; Talbi, 2009) . Son utilizados, tradicionalmente, para resolver problemas de búsqueda y optimización, en los que se tiene que determinar la mejor solución dentro de un conjunto de soluciones factibles (Gil, 2006) . Este método está basado en el proceso genético de los organismos vivos, en el cual se espera que a lo largo de las generaciones, las poblaciones evolucionen de acuerdo con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin en el año 1859 (Abbass et al., 2002; Moujahid et al., sf).

Los algoritmos genéticos forman parte de los denominados *métodos evolutivos de poblaciones*, ya que a diferencia de otros métodos en los que en cada iteración se tiene una única solución que está cambiando con el objetivo de explorar el espacio de búsqueda,

en los algoritmos genéticos se trabaja con un conjunto de posibles soluciones dentro del espacio de búsqueda denominadas instancias. A este conjunto de soluciones se le denomina *población de instancias*. De este modo, en lugar de pasar de una solución única a la siguiente, se pasa de una población de instancias a otra mediante la combinación de varias soluciones. Básicamente, hay dos modelos poblacionales que se utilizan comúnmente en los algoritmos de esta índole, los modelos de estado estable y los modelos generacionales (Abbass et al., 2002).

Los modelos de estado estable, son aquellos que permiten que instancias de la generación actual sobrevivan para formar parte de la siguiente generación, mediante la eliminación de instancias que se consideran menos aptas y las mejores sobreviven a la generación siguiente. Mientras que los modelos generacionales construyen nuevas poblaciones de cero usando información de la población anterior pero eliminándola por completo (Abbass et al., 2002) ; de esta manera, las generaciones son desechadas y reconstruidas por completo en cada iteración.

Aunque existen métodos diversos para la construcción de la población inicial de instancias, la forma más frecuente de hacerlo es mediante una generación aleatoria (Talbi, 2009) . Esto es, escoger un conjunto al azar de soluciones factibles del espacio de búsqueda que serán consideradas en la primera generación (de los Cobos et al., 2010; Mitchell, 1999).

Adaptabilidad de una instancia (fitness)

Cada uno de las instancias dentro de una generación deberá tener asociado un valor de ajuste o valor de bondad, también llamado *aptitud* o *fitness* por el idioma inglés. Este valor deberá cuantificar la calidad de la solución para el problema a resolver, de forma que se pueda comparar soluciones dentro de la misma generación, así como de generaciones anteriores (Gestal, sf) . El valor de aptitud debe permitir realizar un completo

ordenamiento de todas las soluciones del espacio de búsqueda (Talbi, 2009) , de manera que se pueda discernir para cualquier par de soluciones factibles, cual solución es mejor según el problema en estudio. Sin embargo, este mismo autor indica que, a menudo, el cálculo de dicho valor es la parte más cara de un algoritmo, siendo así un proceso considerado computacionalmente costoso.

Para los problemas de optimización, el valor de aptitud, para una solución dada, está relacionado con la función objetivo del problema, f. Así, si el problema es de maximización, entonces entre dos soluciones factibles, será mejor solución aquella que tenga el valor más alto en la función objetivo, la cual será tomada como fitness. En caso de que el problema sea de minimización, será mejor solución aquella cuyo valor en la función objetivo sea menor, para los cuales, se puede tomar como valor de aptitud el recíproco de la función objetivo $1/f$, siempre que dicha función sea positiva, o bien en caso de no serlo, su valor de aptitud puede tomarse como el inverso aditivo de la función objetivo, $-f$. De manera que se pueda garantizar que a mayor valor de aptitud, mejor solución para el problema a resolver.

Por ende, en el algoritmo genético para clasificación, así como en las demás heurísticas, la función de aptitud o fitness será tomado como el recíproco de la inercia intra-clase. Este permitirá determinar el nivel de adaptabilidad (calidad) de las soluciones del problema.

Caracterización de los algoritmos genéticos

Los algoritmos evolutivos son algoritmos que imitan el comportamiento natural de las especies en nuestro planeta. Existen, básicamente, dos características fundamentales que diferencian a los métodos evolutivos, incluidos en éstos los algoritmos genéticos, de otros métodos computacionales de búsqueda (de los Cobos et al., 2010) :

- Parten de una población inicial de soluciones factibles del problema a resolver o instancias, generalmente construida de forma aleatoria.

- Existe comunicación e intercambio de información entre las instancias de la población. Para el caso específico de los algoritmos genéticos, dicho intercambio se realiza mediante el operador de cruce, que será definido posteriormente.

A partir de una población inicial de soluciones creada, muchas veces en forma aleatoria, y codificada con una representación cromosómica sintética, los algoritmos genéticos son capaces de ir creando soluciones factibles a problemas concretos del mundo real (Gil, 2006) . Se espera que mediante la imitación de los procesos naturales de cruce y mutación con el pasar de las iteraciones, la población de instancia explore el espacio de búsqueda y evolucione a regiones del espacio con alta adaptabilidad.

En la naturaleza, los miembros de una población compiten por recursos, comida, agua y refugio (Gil, 2006) . Incluso miembros de un mismo grupo compiten entre ellos por aparearse, siendo los mejor adaptados a su medio ambiente los que tengan mayor probabilidad de conseguir los recursos y de procrear un número mayor de descendientes, los cuales deberán heredar en alguna medida los genes de los progenitores (Moujahid et al., sf) .

La combinación de buenas características provenientes de diferentes ancestros, puede a veces producir descendientes mejor adaptados que sus ancestros. De esta manera, la especie evoluciona logrando la obtención de características que la hacen mejor adaptada al entorno en el que viven (Gil, 2006) . La aparición de miembros super adaptados puede también ser resultado de mutaciones. En biología, una mutación es el proceso por el cual un alelo de un gen cambia y se convierte en otro distinto, donde en ocasiones dichos cambios pueden producir alteraciones en el fenotipo (Griffihs et al., 2000) .

Los proceso de mutación que suceden con frecuencia durante la reproducción celular y la replicación del código genético son las causantes que los descendientes adquieran características distintas a las que poseen ambos progenitores. Sin mutaciones la vida no podría evolucionar en organismos más complejos y mejor adaptados a su medio circundante. Por esta razón, los procesos de mutación son considerados primordiales para el proceso de evolución de las especies.

De este modo, y como pasa en la naturaleza, se espera que las nuevas generaciones estén mejor adaptadas a su medio que sus antecesores, y es este proceso a lo que se le denominó evolución. Los algoritmos genéticos tratan de realizar una analogía de la evolución, que permita determinar buenas soluciones para problemas de la realidad, siempre que se haga una codificación eficiente y exista una medida de calidad para cada una de las soluciones. Buscando que los individuos menos adaptados y sus rasgos desaparezcan en el tiempo, mientras que los individuos con mayor adaptabilidad y sus características perduren en las generaciones futuras.

Sobre la condiciones de convergencia de algoritmo genético, en Rudolph (1994) se analiza las propiedades de convergencia del algoritmo genético canónico. Este autor demuestra, por medio de un análisis de cadena de Markov finita homogénea, que un algoritmo genético canónico no tendrá certeza de convergencia hacia el óptimo global independientemente de la inicialización, operador de cruce y la función objetivo que se posean. También demuestra que las variantes del algoritmo que mantengan siempre la mejor solución en la población (elitismo), ya sea antes o después de la selección, siempre convergen al optimo global. La demostración de este resultado puede ser consultada en Rudolph (1994).

Codificación

El primer detalle que se debe resolver para aplicar un algoritmo genético a un problema real es determinar una codificación cromosómica de las soluciones factibles. A cada posible solución del problema en cuestión deberá asignársele un código único. En general, una representación ha de ser capaz de identificar las características constituyentes de un conjunto de soluciones, de forma que distintas representaciones dan lugar a distintas perspectivas y por tanto distintas soluciones (Gil, 2006) .

La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas (Gil, 2006; Lee & El-Sharkawi, 2008) , por lo que es de suma importancia estudiar las soluciones de manera que se pueda realizar una adecuada y eficiente codificación.

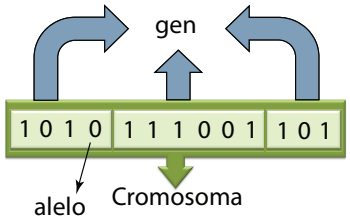


Figura 1.1: Representación cromosómica de una solución con tres genes, cada uno con una codificación binaria de diferente tamaños.

Cualquier solución potencial a un problema dado, puede ser representada asignando valores a una serie de parámetros. El conjunto de todos los parámetros necesarios para representar una solución se codificará en una cadena de caracteres denominada cromosoma (Gestal et al., 2010; Moujahid et al., sf) . A cada parámetro se le llama gen, y cada gen está codificado por una subcadena de caracteres. Cada uno de los caracteres que conforma los genes se le denomina alelo (Mitchell, 1999) . En la figura 1.1 se puede apreciar la representación de un cromosoma con tres genes, cada gen con una codificación binaria de

diferentes tamaño.

El conjunto de los parámetros representado por un cromosoma particular recibe el nombre de genotipo. El genotipo contiene la información necesaria para la construcción del organismo, en este caso una de las soluciones factibles del problema en cuestión. La solución del problema que resulta de la información del genotipo se denomina fenotipo (Gestal et al., 2010) .

Usualmente, los caracteres que conforman el genotipo se expresan en código binario, es decir con ceros y unos (ver figura 1.1). Sin embargo, es posible usar codificaciones con números enteros, reales o incluso letras del alfabeto según la necesidad (Gestal et al., 2010; Gil, 2006; Lee & El-Sharkawi, 2008) . Más aún, cada gen, en un cromosoma, puede poseer diferente codificación; por ejemplo, la figura 1.2 representa un cromosoma compuesto por cuatro genes de diferentes longitudes y con codificación binaria, entera, real y alfabetica respectivamente.

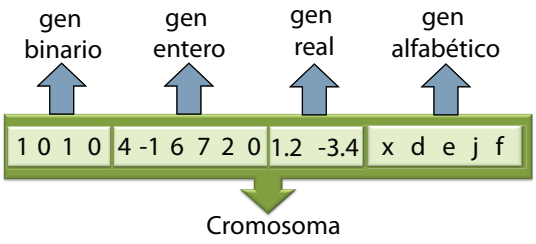


Figura 1.2: Representación de un cromosoma con cuatro genes de diferente longitud y con codificación diferente en cada uno de los genes.

Operadores de los Algoritmos Genéticos

Parte fundamental de la analogía de la evolución genética son los operadores que ayudan en la imitación del proceso. El cruce, la mutación y la selección son los principales operadores

aplicados en los algoritmos genéticos simples (Abbass et al., 2002; Mitchell, 1999) . El cruce se encarga de generar la descendencia, por medio de la combinación de características de dos padres; la mutación garantiza la variabilidad que debe existir en la población; y la selección se encarga de que los miembros más adaptados y sus genes perduren en las generaciones futuras. A continuación se detallan más a fondo estos operadores.

- **Selección**

  Este operador realiza una selección de miembros o instancias que conformarán la nueva generación, imitando la selección natural de las especies. Es esencial que la selección respete la regla de que los individuos más adaptados tengan mayor probabilidad de ser seleccionados para conformar la nueva generación, o bien para ser cruzados durante el operador de cruce. Sin embargo, al igual que pasa en la naturaleza, los individuos menos aptos también deben tener la posibilidad de ser seleccionados y de reproducirse (de los Cobos et al., 2010) .

  Existen diversas formas de realizar el proceso de selección, entre los cuales el método de selección directa y el método de selección estocástica son los más comunes (ver Abbass et al. (2002); Peinado et al. (2003); Valvert (2006) para mayor detalle). El proceso de selección directa responde a criterios de selección del tipo "los k mejores" o "los k peores", mientras que entre los métodos de selección estocástica se encuentran:

  - **Selección por sorteo o ruleta simple:** Consiste en la construcción de una ruleta circular, cuya circunferencia mide igual a la suma del valor de aptitud de todas las instancias en la población que será sometida al proceso de selección. Se coloca cada instancia de la población en un sector circular de la ruleta cuyo segmento circular mide igual a su propio valor de aptitud, ver figura 1.3.

    Para seleccionar una instancias se hace girar la ruleta, seleccionando así la instancia que señale el marcador al detenerse el movimiento. De este modo,

cada escogencia es independiente de las demás y las probabilidades de una instancia de ser escogida no cambia de una selección a otra. Además, por cada instancia que se debe seleccionar, es necesario hacer girar la ruleta una vez.

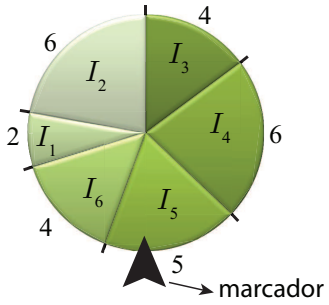


Figura 1.3: Ejemplo de una ruleta simple para 6 instancias I_1, I_2, I_3, I_4, I_5 e I_6 con fitness respectivo 2, 6, 4, 6, 5 y 4.

Gestal et al. (2010) indica que es un método muy sencillo pero ineficiente a medida que aumenta el tamaño de la población (su complejidad es $O(n^2)$). Presenta además el inconveniente de que el peor individuo puede ser seleccionado más de una vez, y el mejor individuo podría no ser seleccionado.

- **Selección por escaños:** Se seleccionan las instancias según la ley de d'Hont la cual consiste en una fórmula electoral que permite repartir a las candidaturas los cargo electorales disponibles, en proporción a los votos emitidos.

  Para entender el algoritmo de la selección por escaños, considérese el siguiente ejemplo:

  Suponga que se tiene cinco instancias I_1, I_2, I_3, I_4 e I_5 cuyos valores de aptitud corresponden a 340, 280, 160, 60 y 15 respectivamente. Suponga que se desea seleccionar cuatro instancias. La repartición, según la ley de d'Hont consiste en los siguientes pasos:

  - *Primera escogencia:* se selecciona la instancia cuyo valor de aptitud sea más alto. De este modo la instancia I_1 es seleccionada. A esta instancia se

le asigna, temporalmente, un valor de aptitud de $340/2 = 170$. Las demás se mantienen igual. En la tabla 1.2 se presentan los valores de aptitud asignados luego de la primera escogencia.

Tabla 1.2: Instancias y fitness asignados en la primera iteración.

Intancias	I_1	I_2	I_3	I_4	I_5
Fitness	170	280	160	60	15

- *Segunda escogencia:* se selecciona la instancia con valor de aptitud más alto. De este modo la instancia I_2 es seleccionada. A esta instancia se le asigna, temporalmente, un valor de aptitud de $280/2 = 140$. Las demás se mantienen igual. En la tabla 1.3 se presentan los valores de aptitud asignados luego de la escogencia.

Tabla 1.3: Instancias y fitness asignados en la primera iteración.

Intancias	I_1	I_2	I_3	I_4	I_5
Fitness	170	140	160	60	15

- *Tercera escogencia:* se selecciona la instancia con valor de aptitud más alto. De este modo la instancia I_1 es seleccionada nuevamente. A esta instancia se le asigna, temporalmente, un valor de aptitud de $340/3 = 113.\overline{3}$, que corresponde a un tercio del valor original pues ya se ha seleccionado dos veces. Las demás se mantienen igual. En la tabla 1.4 se presentan los valores de aptitud asignados luego de la escogencia.

Tabla 1.4: Instancias y fitness asignados en la primera iteración.

Intancias	I_1	I_2	I_3	I_4	I_5
Fitness	$113.\overline{3}$	140	160	60	15

- *Cuarta escogencia:* se selecciona la instancia con valor de aptitud más alto. De este modo la instancia I_3 es seleccionada. A esta instancia se le asigna, temporalmente, un valor de aptitud de $160/2 = 80$, pues es la primera vez que ha sido seleccionada. Las demás se mantienen igual. En la tabla 1.5 se presentan los valores de aptitud asignados luego de la escogencia.

Tabla 1.5: Instancias y fitness asignados en la primera iteración.

Intancias	I_1	I_2	I_3	I_4	I_5
Fitness	$113.\overline{3}$	140	80	60	15

El proceso da como resultado la selección de las instancias I_1, I_2 e I_3, siendo la primera seleccionada dos veces.

- **Por torneo:** Según Gestal et al. (2010) , existen dos versiones de este método de selección, el torneo determinístico y el torneo probabilístico. En ambos se toman al azar un número p de instancias (usualmente se escoge $p = 2$), el torneo determinístico consiste en seleccionar el miembro más adaptado el grupo tomado, mientras que en el torneo probabilístico se selecciona la instancia por medio de una ruleta dando mayor probabilidad al miembro más adaptado del grupo tomado.

- **Cruce:**

Generalmente, este operador es binario lo que significa que requiere de dos argumentos para poderse aplicar (en este caso dos instancias). Su principal tarea es combinar las características de los padre para generar la descendencia. El operador de cruce depende en gran medida de la codificación cromosómica realizada del problema (Talbi, 2009) .

Los operadores de cruce utilizados con mayor frecuencia corresponden a cruce basados en un punto de corte, cruce basado en dos puntos de corte y cruce uniforme;

mismos que son detallados a continuación:

- *Cruce basado en un punto de corte:* Este es conocido como el operador de cruce clásico. Una vez escogidos los dos padres o progenitores, se cortan sus cromosomas en un punto al azar (mismo punto para ambos padres) generando así dos segmentos en cada cromosoma. Luego se intercambian los segmentos obtenidos para generar la descendencia que hereda información genética de ambos padres (Abbass et al., 2002; Gestal et al., 2010; Gil, 2006; Moujahid et al., sf) . En la figura 1.4 se puede observar el cruce basado en un punto de corte.

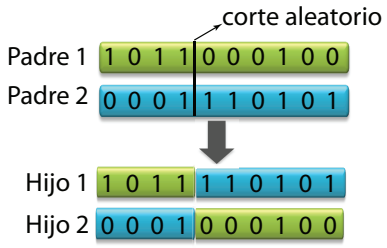


Figura 1.4: Cruce basado en un punto de corte.

- *Cruce basado en dos puntos de corte:* corresponde a una generalización del cruce basado en un puntos de corte. Esta vez se escogen, de manera aleatoria, dos punto donde cortar el cromosoma de cada padre (los mismo puntos de corte para ambos padres), de manera que de cada cromosoma se generen tres segmentos del código. Luego se intercambian el segmento central de uno de los padres con los segmentos laterales del otro (Abbass et al., 2002; Gestal et al., 2010; Gil, 2006; Moujahid et al., sf) . En la figura 1.5 se puede observar el cruce basado en dos puntos de corte.

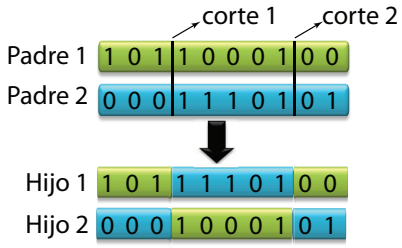


Figura 1.5: Cruce basado en dos puntos de corte.

- *Cruce uniforme:* consiste en realizar un test aleatorio para decidir de cual de los progenitores se toma cada alelo de la cadena del cromosoma del descendiente (Gil, 2006) . Es decir, la construcción del cromosoma de la descendencia se realiza alelo por alelo, escogiendo cada uno, por medio de una regla probabilistica, de un padre u otro. La regla de asignación de alelos puede diseñarse de manera que beneficie al padre mejor adaptado.

- **Mutación:**

  La mutación es un operador unitario de la población, lo que significa que requiere un único argumento para aplicarse. Consiste en la realización de pequeños cambios en los alelos de algunos cromosomas específicos Talbi (2009) . La introducción de pequeñas variaciones dentro del material genético mantiene suficiente variabilidad dentro de la población, evitando la convergencia prematura. La mutación también garantiza que toda región del espacio de búsqueda es potencialmente estudiada.

  Generalmente, la mutación consiste en cambiar un alelo escogido aleatoriamente por un valor válido tomado al azar.

Método de paro

El método de paro de un algoritmo es una condición cuya verificación detiene el proceso y obliga al algoritmo a retornar la mejor solución obtenida o un mensaje de fracaso en caso de haber evidencia de fallo en la búsqueda.

Según Gestal et al. (2010) entre los criterios de parada más usuales están, detener el algoritmo cuando:

- Las mejores instancias de la población representan soluciones suficientemente buenas del problema a resolver.
- La población ha convergido, esto es, cuando el 95 % de la población de instancias posee el mismo valor de aptitud.
- Sea alcanzado el número máximo de generaciones especificado.

Por su parte propone que el criterio de parada, generalmente viene determinado por criterios a priori sencillos como por ejemplo:

- un número máximo de generaciones,
- un tiempo máximo de resolución,

o más eficientemente por estrategias relacionadas con indicadores del estado de evolución de la población, como son:

- la pérdida de diversidad dentro de la población,
- no haber mejora en un cierto número de iteraciones,

siendo por lo general una condición mixta la más utilizado, es decir, limitar el tiempo de ejecución a un número de iteraciones y tener en cuenta algún indicador del estado de la población para considerar la convergencia.

Algoritmo genético simple

Los algoritmos genéticos simples son aquellos que poseen al menos los tres operadores antes descritos. Si bien existen otras versiones en las cuales se incorporan otros operadores, todos deben contar con el operador de selección, cruce y mutación. Finalmente, en el algoritmo 1 se muestra el pseudocódigo de un algoritmo genético simple.

Algoritmo 1 Algoritmo genético simple

Entrada: Parámetros de entrada.

Salida: El individuo mejor adaptado (mejor solución obtenida).

1: Generar una población inicial Ψ, con m individuos.
2: Calcular el fitness de cada individuo en Ψ.
3: Contador $\leftarrow 0$.
4: **mientras** no haya sido satisfecho el criterio de parada **hacer**
5: Contador $\leftarrow$ Contador $+1$
6: $\Psi \leftarrow$ Cruces(Ψ)
7: $\Psi \leftarrow$ Mutación(Ψ)
8: $\Psi \leftarrow$ Selección(Ψ)
9: **fin mientras**
10: **retornar** $I_0 \in P$, donde I_0 es la mejor solución obtenida.

1.3.3. Enjambres de partículas

La heurística de optimización basada en enjambres de partículas, o *particle swarm* en inglés, es una técnica computacional evolutiva, inspirada en el comportamiento social de los enjambres, tales como las parvadas de aves, los enjambres de insectos y los cardúmenes de peces, durante el proceso de exploración en búsqueda de alimento y refugio.

La imitación de los enjambres para la resolución de problemas de búsqueda fue propuesta por Eberhart y Kennedy en el año de 1995 (Eberhart & Kennedy, 1995) , durante una investigación en la que trataban de simular los movimientos sincronizados de enjambres de pájaros, como parte de un estudio socio-cognitivo que investigaba la noción de la "inteligencia colectiva".

El concepto surge del hecho de que la capacidad de un enjambre sobrepasa la capacidad de cualquiera de sus individuos, en este sentido Kennedy & Eberhart (2001) indica que una bandada de aves tiene propiedades más allá de las propiedades de las propias aves. De igual manera establece el hecho de que un insecto puede tener sólo unos pocos cientos de células cerebrales, pero las organizaciones de insectos son capaces de maravillas arquitectónicas, sistemas de comunicación elaborados y excelente resistencia a las amenazas de la naturaleza.

Esta técnica tiene semejanza con ciertos métodos de optimización heurísticos tales como algoritmos genéticos y colonias de hormigas, los cuales son métodos poblacionales, en el sentido que parten de una población inicial, la cual está formada por soluciones al azar del problema a resolver. Mediante la interacción entre los individuos de la población y el intercambio de información básica, dichos individuos exploran el espacio de búsqueda, evolucionando con el pasar de las iteraciones a una solución suboptimal.

En los algoritmos genéticos, los individuos son integrantes de una generación, los cuales

interactúan entre sí mediante el operador de cruces, en el cual se realiza el intercambio de información. En el caso de la optimización basada en colonias de hormigas, los individuos de la población son hormigas artificiales que comparten información por medio de un rastro de feromona. Finalmente, en la metaheurística de optimización por enjambres de partículas, los individuos son partículas en un espacio multidimensional que se encuentran en movimiento, en búsqueda de buenas soluciones (de los Cobos et al., 2010; Eberhart & Kennedy, 1995) .

Descripción del método

El comportamiento social de algunos grupos de individuos permite que un subconjunto de ellos posea comportamientos sincronizados para lograr un objetivo común, para el bienestar del enjambre entero. Por ejemplo, algunos tipos de aves, durante la emigración de largas distancias, vuelan en formación y con una sincronización casi perfecta, con el único objetivo de ahorrar energía durante el vuelo. El ave que encabeza la formación debe realizar más esfuerzo que las demás, sin embargo, es la que guía a las otras aves en su ruta, definiendo la dirección, la velocidad y los movimientos que las otras aves deben imitar.

De una manera similar a las aves, la optimización por enjambres de partículas consiste en un enjambre artificial de individuos, los cuales son llamados partículas, en el que cada uno de ellos debe moverse por el espacio multidimensional, en búsqueda de soluciones suboptimales (Lima & Barán, 2006; Sedighizadeh & Masehian, 2009) . Los movimientos de estos agentes de búsqueda no son aleatorios, sino que se basan en los principios de *evaluación*, *comparación* e *imitación* (Xie et al., 2002a) , los cuales se presentan a continuación (de los Cobos et al., 2010; Kennedy & Eberhart, 2001) :

- **Evaluación:**

  Los organismos vivos poseen como una de las principales características la tendencia a evaluar los estímulos del medio ambiente que les rodea. El aprendizaje no puede ocurrir sin que los organismos posean la capacidad de evaluar. De esta manera, el aprendizaje puede concebirse como todo cambio que permite que el organismo mejore la evaluación media de su ambiente circundante. Los integrantes de un grupo de individuos deben ser capaces de evaluar su propio estado, ver si han mejorado o empeorado con el pasar del tiempo, o bien percibir si el lugar donde se encuentra en la actualidad es mejor o peor que otros lugares en donde han estado.

- **Comparación:**

  En casi todo lo que pensamos y hacemos, nos evaluamos a nosotros mismos mediante la comparación con otros, ya sea en la evaluación de nuestras actividades, nuestro estatus social, personalidad, inteligencia u otros aspectos de opinión y de capacidad. La comparación de un individuo con los demás de una sociedad es una actividad que propicia la construcción de los estándares sociales. El comportamiento social de los individuos dentro de una sociedad es construido por medio de la comparación con otros individuos, imitando así aquellos que son mejores que nosotros mediante un estándar social. En enjambres de partículas, cada partícula debe compararse con sus vecinos como una medida crítica e imitar solamente aquellos vecinos que son superiores.

- **Imitación:**

  Es concebible pensar que la imitación podría darse en todas partes en la naturaleza, como un método efectivo de aprendizaje de actividades nuevas. La verdadera imitación es fundamental para la sociabilidad humana, y es fundamental para la adquisición y mantenimiento de las capacidades mentales de la especie, así como para la adaptabilidad de los organismos a un nuevo entorno.

Los principios de evaluar, comparar e imitar se presentan simultáneamente en algunos individuos, inclusive en los organismos socialmente simplificados. La implementación de dichos principios en algoritmos de búsqueda pueden ayudar a resolver problemas complejos, ya que dichos principios facultan al método a realizar adaptaciones a su espacio de trabajo (Kennedy & Eberhart, 2001) .

Modelación matemática

Considere un problema de optimización en el cual se pretende minimizar una función $f : \Omega \to \mathbb{R}$, donde Ω es el conjunto de soluciones factibles del problema, tal que $\Omega \subset \mathbb{R}^N$. El método de optimización por enjambres de partículas consiste en considerar un vector $\mathcal{E}$ de m soluciones factibles en Ω que se denominará *enjambre* y en el cual a cada solución se le conocerá con el nombre de *partícula*. Es decir,

$$\mathcal{E} = \left(\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \ldots, \mathcal{P}_m\right),$$

donde $\mathcal{P}_k \in \Omega$ para $k = 1, 2, 3, \ldots, m$.

Las partículas, actualmente estáticas, deberán ser impulsadas en cada iteración. La velocidad y la dirección del movimiento de cada partícula deben incluir los tres principios de evaluación, comparación e imitación. Para esto, es necesario que cada partícula $\mathcal{P}_k$ conozca

- **Su posición actual**

  Cada partícula $\mathcal{P}_k$ debe tener un vector de posición

$$\overrightarrow{\chi_k} = (\chi_{k1}, \chi_{k2}, \chi_{k3}, \ldots, \chi_{kN}),$$

el cual corresponde a las coordenadas que indique la posición de la partícula en el conjunto de soluciones factibles Ω.

- **La velocidad con la cual ha llegado a su posición actual**

  Cada partícula tiene asociado un vector de velocidad:

$$\overrightarrow{\mathbf{v}_k} = (v_{k1}, v_{k2}, v_{k3}, \ldots, v_{kN}),$$

  el cual registra la velocidad con la que dicha partícula ha llegado a la posición actual $\overrightarrow{\chi_k}$.

- **Memoria de la mejor solución encontrada por la partícula**

  Cada partícula debe recordar la mejor posición en la que ha estado. Esta posición se denotará por:

$$\overrightarrow{\chi\mathbf{p}_k} = (p_{k1}, p_{k2}, p_{k3}, \ldots, p_{kN}).$$

- **Conocimiento de la mejor posición encontrada por el enjambre**

  El intercambio de información entre los individuos del enjambre es de vital importancia. Cada partícula debe conocer la mejor posición global del enjambre encontrada por cualquiera de las partículas. Dicha posición será denotada por:

$$\overrightarrow{\chi\mathbf{m}} = (m_1, m_2, m_3, \ldots, m_N).$$

Tal y como ya se mencionó, la propulsión de las partículas debe regirse por los tres principios de evaluación, comparación e imitación. Para esto, se debe considerar toda la información que la partícula conoce, tanto de su propia memoria como de la parte social del enjambre.

De este modo, para actualizar la posición $\overrightarrow{\chi_k}^{(t)}$ de la partícula $\mathcal{P}_k$ en el tiempo t

(iteración t), se utilizará el siguiente proceso de actualización, el cual se puede revisar con más detalle en (Bratton & Kennedy, 2007; Kennedy & Eberhart, 2001; Xie et al., 2002a).

La nueva posición de la partícula $\mathcal{P}_k$ será:

$$\overrightarrow{\chi_k}^{(t+1)} \leftarrow \overrightarrow{\chi_k}^{(t)} + \overrightarrow{\mathbf{v}_k}^{(t+1)},$$

donde cada componente del vector $\overrightarrow{\mathbf{v}_k}^{(t+1)}$ está dada por:

$$v_{kj}^{(t+1)} = \alpha v_{kj}^{(t)} + \kappa_1[p_{kj}^{(t)} - x_{kj}^{(t)}] + \kappa_2[m_k^{(t)} - x_{kj}^{(t)}],$$

con

$$j = 1, 2, 3, \ldots, N \; ; \quad \kappa_1 = c_1 \cdot \mathrm{rnd}(0,1) \quad \text{y} \quad \kappa_2 = c_2 \cdot \mathrm{rnd}(0,1);$$

donde $\mathrm{rnd}(0,1)$ denota un número generado aleatoriamente en $]0,1[$, no siendo necesariamente el mismo para κ_1 y κ_2. Además, α, c_1 y c_2 son parámetros que se deben ajustar y que tienen las siguientes funciones:

- α: es un parámetro de tendencia que regula la influencia que posee la velocidad $\overrightarrow{\mathbf{v}_k}$ anterior en el cálculo de la nueva velocidad.
- c_1: es un parámetro individual que se encarga de regular la influencia máxima que posee la mejor experiencia de la partícula (mejor solución encontrada), para el cálculo de su nueva velocidad $\overrightarrow{\mathbf{v}_k}^{(t+1)}$.
- c_2: es un parámetro social que representa la influencia máxima que posee la mejor experiencia encontrada por el enjambre completo.

Lima & Barán (2006) y Sedighizadeh & Masehian (2009) indican que típicamente en sus investigaciones se asignan $\alpha = 0.8$ y $c_1 = c_2 = 2$. Por su parte, Hassan et al. (2005)

propone algunos rangos para dichos parámetros, los cuales se indican a continuación: $\alpha \in [0.4, 1.4]$, $c_1 \in [1.5, 2]$ y $c_2 \in [2, 2.4]$. Este autor indica que en su investigación (ver Hassan et al., 2005) encontró que el ajuste del peso de los tres factores α, c_1 y c_2 en 0.5, 1.5 y 1.5, respectivamente, proporciona la mejor velocidad de convergencia para todos los problemas de prueba considerados en su trabajo.

Algunos autores han puesto mucho empeño en determinar condiciones para la convergencia y divergencia del método en cuestión, con respecto a esto, en su trabajo Clerc (1999) modela las interacciones del enjambres partículas utilizando un conjunto de ecuaciones lineales complejas, el cual le permite determinar condiciones de convergencia y divergencia del método, sin embargo aún no existe ninguna prueba de la existencia de condiciones que garanticen la convergencia al óptimo global del problema.

Bratton & Kennedy (2007) resume la condición establecidas por Clerc en su trabajo, en donde agrega un factor Z en el cálculo de la velocidad, denominado factor de constricción, el cual está definido por

$$Z = \frac{2}{\left|2 - w + \sqrt{w^2 - 4w}\right|},$$

donde $w = c_1 + c_2$. Clerc asegura que si $w < 4$, el enjambre se mueve lentamente en espiral alrededor de la mejor solución encontrada en el espacio de búsqueda sin ninguna garantía de convergencia, mientras que para $w > 4$ la convergencia del método sería rápida y garantizada, sin embargo, no hay certeza de que la convergencia sea al óptimo global del problema.

Sin embargo, es importante estudiar más a fondo el ajuste de parámetros en el método de optimización por enjambres de partículas, de forma que exista alguna evidencia de que un conjunto de parámetros es significativamente mejor que cualquier otro conjunto.

Método de paro

Como método de paro para el algoritmo se ha sugerido, por varios autores, detener el programa cuando un máximo de iteraciones se ha ejecutado (Alias et al., 2011; Lima & Barán, 2006; Xie et al., 2002b) . Por su parte Hassan et al. (2005) propone un método de paro que consiste en iterar hasta que la mejora del fitness, en las últimas q iteraciones, no mejore significativamente, la mejor solución encontrada por el enjambre hasta el momento.

Finalmente, en el algoritmo 2 se presenta el pseudocódigo de esta técnica de optimización.

Algoritmo 2 Algoritmo de optimización por enjambres de partículas

Entrada: Parámetros de entradas α, c_1, c_2.
Salida: La mejor posición encontrada por el enjambre.

1: Construye aleatoriamente el enjambre $\mathcal{E} = (\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m)$
2: Contador $\leftarrow 0$.
3: **mientras** no haya sido satisfecho el criterio de parada **hacer**
4: Evaluar el costo de cada partícula: $W(\mathcal{P}_k)$.
5: **para** $k \leftarrow 1$ **hasta** m **hacer**
6: Actualizar la velocidad de la $k-$ésima partícula $\overrightarrow{\mathbf{v}}_k^{(t+1)}$:

$$v_{kj}^{(t+1)} \leftarrow \alpha v_{kj}^{(t)} + \kappa_1[p_{kj}^{(t)} - x_{kj}^{(t)}] + \kappa_2[m_j^{(t)} - x_{kj}^{(t)}]$$

7: Mover la $k-$ésima partícula a su nueva posición $\overrightarrow{\chi}_k^{(t+1)}$.

$$\overrightarrow{\chi}_k^{(t+1)} \leftarrow \overrightarrow{\chi}_k^{(t)} + \overrightarrow{\mathbf{v}}_k^{(t+1)}$$

8: Actualizar la mejor posición de la $k-$ésima partícula en caso de mejorar la existente. Actualizar $\overrightarrow{\chi\mathbf{p}_k}$.
9: Actualizar la mejor posición encontrada por el enjambre en caso de mejorar la existente. Actualizar $\overrightarrow{\chi\mathbf{m}}$.
10: **fin para**
11: Contador $\leftarrow$ Contador $+ 1$.
12: **fin mientras**
13: **retornar** La mejor solución encontrada durante la búsqueda

1.3.4. Búsqueda tabú

La búsqueda tabú es un procedimiento utilizado para resolver problemas de optimización y ha sido empleado en una amplia variedad de problemas clásicos. Fue introducido por Fred Glover en dos artículos en 1989 y 1990 (Glover, 1989, 1990), y en sus primeros años se limitó a resolver problemas en el área de la optimización combinatoria, tales como el problema del agente viajero y problemas de grafos. No obstante, a partir de los años 90 se han generado aplicaciones a problemas más complejos (Lee & El-Sharkawi, 2008).

Esta técnica permite la exploración de regiones de factibilidad de difícil exploración, basando su potencial en el uso de una memoria flexible. Es decir, explota la información histórica guardando en una memoria variante y de corto plazo, una cierta cantidad de información sobre las soluciones ya encontradas por el algoritmo. Esto con el objetivo de no repetir la exploración en un futuro cercano y permitirle buscar en otras regiones del espacio de soluciones factibles. La razón de esta memoria es excluir los movimientos que pueden regresar al algoritmo a algún punto ya explorado en una iteración anterior.

La búsqueda tabú emplea la memoria con el objetivo de dirigir la búsqueda considerando la historia reciente de la exploración. En general, extrae información de lo ya acontecido en las iteraciones recientes para tomar decisiones en función de ello. En este sentido puede decirse que hay un cierto aprendizaje y que la búsqueda es inteligente (Riojas, 2005).

Similarmente, la memoria tiene la funcionalidad de evitar ciclos, generados en muchos casos por un óptimo local con mucha atracción, incorporando los últimos movimientos realizados como *movimientos o restricciones tabú*. Esta condición tabú puede ser ignorada bajo determinadas circunstancias, dando lugar a los llamados *criterios de aspiración*. Dichos criterios permiten aceptar una solución, aunque halla sido declarada tabú, siempre que ésta mejore cualquier otra previamente encontrada.

De manera similar a como lo hacen otras heurísticas, como el caso de sobrecalentamiento simulado y aceptación de umbrales, la búsqueda tabú permite hacer una búsqueda aceptando nuevas soluciones que no necesariamente sean tan buenas como la actual, pero esta eventual "pérdida de calidad" es la que le permite al algoritmo escapar de óptimos locales.

Según se explica en de los Cobos et al. (2010), la búsqueda tabú se basa en tres puntos principales: primero, el uso de estructuras de memorias basadas en atributos diseñados para permitir criterios de evaluación e información de búsqueda histórica; segundo, un mecanismo asociado de control mediante el empleo de estructuras de memoria (restricciones tabú y criterio de aspiración); y, tercero, la incorporación de funciones de memoria de diferentes lapsos (memoria de largo y de corto plazo).

En este método, en cada iteración se selecciona el mejor elemento en el vecindario de la solución actual, el cual se convierte en la nueva solución actual. La diferencia con los algoritmos tradicionales de búsqueda local radica en que el vecindario de una solución x^c cambia durante la ejecución del algoritmo (Winker, 2001). Lo anterior debido a que elementos del espacio de búsqueda que han sido visitados por el algoritmo recientemente, pueden ser eliminados de los vecindarios al darles el estado *tabú* por un cierto número de iteraciones. Según se explica en Winker (2001), la lista tabú tiene que ser usada como una manera de guiar al algoritmo lejos de las regiones del espacio de búsqueda que ya han sido exploradas.

Conceptos relacionados con el método

La lista tabú

Una lista tabú τ es una memoria de corto plazo que almacena información sobre las soluciones que fueron exploradas por el algoritmo en un pasado reciente. También se puede

almacenar información pertinente para el problema a resolver. Por ejemplo, en el problema del agente viajero, la lista puede contener las aristas o vértices (haciendo referencia a la representación mediante grafos del problema), que han sido visitadas por el algoritmo en una cierta cantidad de iteraciones recientes, o bien, puede contener las aristas que del todo no se quiere que sean visitadas por el agente. En de los Cobos et al. (2010) se indica que un movimiento permanece como tabú solo durante un cierto número de iteraciones, dado que para cada movimiento $m \rightarrow m^*$, el movimiento opuesto $m^* \rightarrow m$ se adiciona al final de τ y luego el movimiento más antiguo en τ se elimina, de forma que la lista tabú τ es una lista cíclica.

Las listas tabú son memorias que almacenan la historia reciente de la búsqueda previniendo los ciclos. Entre ellas se pueden encontrar *las memorias de corto plazo* que previenen la reversión del algoritmo hacia movimientos ejecutados recientemente y *las memorias de largo plazo* que más bien tratan de reforzar las soluciones de alta calidad encontradas en iteraciones anteriores (Gendreau & Potvin, 2008) .

Sin embargo, en Gendreau & Potvin (2008) se advierte que no es práctico almacenar soluciones completas en la lista tabú, esto debido a su alto costo computacional y, adicionalmente, pues no permitiría que se aplique un criterio de aspiración. El uso más común para la lista tabú es almacenar las últimas transformaciones que ha sufrido la solución actual y luego prohibir las transformaciones que las reversen.

El tamaño de la lista tabú (*tabu tenure*, en la literatura en inglés) es un parámetro que debe ser ajustado, debido a que si es demasiado pequeño pueden generarse ciclos, pero si es demasiado grande puede restringir la búsqueda al impedir la evasión de óptimos locales con mucha atracción (Murillo, 2000). En general, el tamaño de la lista corresponde al número de iteraciones (movimientos o transiciones) que un atributo se mantiene designado como tabú (Lee & El-Sharkawi, 2008).

Conforme la búsqueda progresa, la forma de la evaluación empleada por la búsqueda tabú llega a ser más adaptativa, incorporando referencias concernientes para la intensificación y la diversificación regional de la búsqueda (de los Cobos et al., 2010). Tanto la intensificación como la diversificación son consideradas como funciones avanzadas de la búsqueda tabú, en el sentido que pueden ser agregadas al algoritmo básico de búsqueda tabú que usan memorias de corto plazo con listas tabú y con un criterio de aspiración. Así, la búsqueda comienza con el algoritmo básico y después se continúa con la intensificación y diversificación (Lee & El-Sharkawi, 2008). Estos procesos se describen con más detalle a continuación, junto con el criterio de aspiración.

Intensificación

En el método de búsqueda tabú, la intensificación consiste en regresar a regiones ya exploradas para estudiarlas más a fondo. Para ello se favorece la aparición de aquellos atributos asociados a buenas soluciones encontradas.

Las memorias de corto plazo permiten identificar elementos de la vecindad del movimiento actual, mientras que en las memorias de intermedio y largo plazo, asociadas con la intensificación, se busca seleccionar soluciones élites (óptimos locales de alta calidad) encontradas en varias ocasiones durante la ejecución del algoritmo (de los Cobos et al., 2010).

La intensificación puede ser implementada comenzando con cualquiera de las soluciones élites encontradas previamente. En este caso, el proceso de intensificación puede ser ejecutado usando básicamente el mismo algoritmo de memoria a corto plazo, haciendo en este caso la búsqueda en un vecindario de una solución élite con el objetivo de encontrar una solución óptima en ese vecindario (Lee & El-Sharkawi, 2008).

Diversificación

La diversificación consiste en visitar nuevas áreas no exploradas del espacio de soluciones. Según se indica en Lee & El-Sharkawi (2008), el objetivo de la diversificación es moverse en el espacio de búsqueda hacia regiones no visitadas, así como evitar ciclos en el algoritmo.

Una forma clásica de diversificación consiste en reiniciar periódicamente la búsqueda desde puntos elegidos aleatoriamente. Si se tiene alguna información acerca de la región factible se puede hacer un "muestreo" para cubrir la región en lo posible, si no, cada vez se escoge aleatoriamente un punto de partida (Riojas, 2005). En este mismo sentido, en Lee & El-Sharkawi (2008) se indica que la diversificación puede ser implementada reiniciando la búsqueda desde una nueva configuración (solución dentro del espacio de búsqueda) o bien modificando las reglas de selección.

Criterio de aspiración

El criterio de aspiración es la condición que permite eliminar a una solución la característica de ser tabú. Es decir, una solución puede llegar a ser aceptada aún cuando ha sido penalizada como tabú, siempre que se satisfaga un cierto umbral de aceptación conocido como criterio de aspiración. En general, se busca analizar si una solución que ha sido asignada como tabú es lo suficientemente buena para relajar dicha restricción (Lee & El-Sharkawi, 2008).

Finalmente, en de los Cobos et al. (2010) se explica que el método de búsqueda tabú puede esbozarse en términos generales, por los siguientes pasos:

1. Encuentre una solución inicial.

2. Construya una estructura de vecindades. Para esto se puede representar a cada

solución factible por medio de un punto en algún espacio y se define un vecindario de cada punto.

3. Escoja el mejor vecino admisible, ya sea que este provoque o no un mejor valor en la función objetivo.

4. Construya la lista tabú y la función de memoria de largo plazo.

5. Actualice las funciones de corto y largo plazo.

6. Realice una intensificación regional de búsqueda.

7. Realice una diversificación regional de búsqueda.

Algoritmo propuesto por Glover

Las notaciones y el algoritmo que se presentarán a continuación son basadas en Glover (1989) y Glover (1990), y corresponden al algoritmo original de búsqueda tabú que fue propuesto por Fred Glover.

Se supone que se quiere minimizar $c(x)$ para $x \in U$, donde U representa el conjunto de posibles soluciones. Se define el conjunto S de todos los movimientos de U en U mediante

$$S := \{s : A \subseteq U \to U, A \neq \emptyset\}.$$

De este modo, un *movimiento* es una aplicación desde un subconjunto no vacío A de U en U. En definitiva, el cambio de una solución inicial a otra solución (es decir, de un elemento de U a otro elemento de U) puede verse como un elemento s de S.

En general, dado un x inicial y otro x' en U, pueden haber varios movimientos $s \in S$ que envían x a x'. El OPT es el procedimiento que se usa para elegir uno de esos s de

manera óptima. A la colección de movimientos que envían x a x' que se descartan en este proceso de selección se le llama *lista tabú* y se le denota mediante τ. A partir de dichas notaciones, Glover propone el algoritmo simple de búsqueda tabú que se especifica en el algoritmo 3.

Además, en Glover (1989) se indica que una escogencia natural realizada por el OPT es seleccionar s tal que

$$c(s) = \text{mín}\{c(s') : s' \in S - \tau\}.$$

De esta manera, cada ejecución de la instrucción 8 en el algoritmo 3, para la elección del s_k, selecciona el movimiento que genere el mejor rendimiento en el valor de la función objetivo, sujeto a la restricción que solo son permitidos movimientos que no estén designados como tabú.

Algoritmo 3 Algoritmo de BT

1: Seleccione una solución inicial $x \in U$.
2: Haga $x^* = x$. // x^* denotará la mejor solución.
3: Inicialice el contador $k = 0$ y haga $\tau = \emptyset$.
4: **si** $S - \tau = \emptyset$ **entonces**
5: Vaya al paso 15.
6: **si no**
7: Haga $k = k + 1$.
8: Seleccione $s_k \in S - \tau$ tal que $s_k = \text{OPT}\{s : s \in S - \tau\}$
9: **fin si**
10: Haga $x = s_k$.
11: **si** $c(x) < c(x^*)$ **entonces**
12: Haga $x^* = x$.
13: **fin si**
14: **si** CantIteraSinMejoras $>$ Cant **ó** $S - \tau = \emptyset$ **entonces**
15: Detener el algoritmo y reportar x^*.
16: **si no**
17: Actualice τ y regrese al paso 4.
18: **fin si**
19: **retornar** x^*.

1.4. Estudios previos en particionamiento

Recientemente han sido propuestas heurísticas de optimización para resolver problemas de optimización discreta y particularmente en el problema de particionamiento de datos. Por esta razón, muchos autores han tratado de encontrar mejores aproximaciones a este problema y empleando dicha estrategia de optimización (Trejos & Murillo, 2004).

El abordaje heurístico a un problema de optimización combinatorio con la complejidad del problema presentado, genera siempre aproximaciones a la solución real. En un tiempo finito y viable, ninguna heurística de optimización, ni ningún algoritmo ofrecerá certeza absoluta de convergencia a la solución óptima. En tal caso, siempre es oportuno explorar el problema, de tal manera que se generen algoritmos heurísticos más robustos y con mayor probabilidad de convergencia hacia la solución óptima, esto con el objetivo paralelo de minimizar el tiempo de ejecución del algoritmo en una computadora.

En Lee & El-Sharkawi (2008) se reafirma esta posición cuando se indica que el uso de estas herramientas ofrece dos ventajas principales que son: el tiempo de ejecución es mucho menor que en los algoritmos tradicionales y que los sistemas modelados con heurísticas son relativamente no sensibles al ruido o los datos aberrantes (*outliers*), lo cual hace que sean modelos más estables.

En particular, según se asevera en Trejos & Murillo (2004), en Costa Rica el grupo de investigadores PIMAD (Programa de Investigación en Modelos y Análisis de Datos) del Centro de Investigación en Matemática Pura y Aplicada (CIMPA) de la Escuela de Matemática de la Universidad de Costa Rica, ha abordado en los últimos años el problema de aplicación de heurísticas modernas de optimización a diversos problemas de Análisis Multivariado de Datos.

En el caso de particionamiento de datos cuantitativos y empleando una métrica euclídea

clásica, diversos autores han implementado técnicas de optimización combinatoria, como sobrecalentamiento simulado, búsqueda tabú y estrategias evolutivas, reportando excelentes resultados e indicando que fueron significativamente mejores que los obtenidos por métodos clásicos como k-medias o el método de Ward (ver Trejos et al. (1998) y Piza et al. (1999)).

Similarmente, en Trejos et al. (2004) se indica que heurísticas como sobrecalentamiento simulado, búsqueda tabú y algoritmos genéticos, han mostrado tener buenas características cuando han sido implementadas en diferentes problemas. Además, añaden que estas heurísticas han sido ampliamente estudiadas por diversos autores en el tema de particionamiento de datos, quienes reportan buenos resultados.

En Trejos et al. (2006) se reporta también una comparación de las metaheurísticas sobrecalentamiento simulado, búsqueda tabú, algoritmos genéticos, colonias de hormigas, enjambres de partículas y los métodos de k-medias y el de Ward, en el tema de particionamiento de datos cuantitativos. Similarmente, en Trejos et al. (1998) se presenta una comparación del porcentaje de atracción hacia el óptimo global al aplicar las heurísticas búsqueda tabú, sobrecalentamiento simulado, algoritmos genéticos y los métodos de k-medias y el de Ward, en tablas clásicas de la literatura del Análisis de Datos, a saber: *French Scholar Notes, Amiard's Fishes, Thomas's Sociomatrix* y *Fisher's Iris*.

Por su parte, en Murillo (2000) se indica la adaptación hecha de la búsqueda tabú al problema de particionamiento de datos. En general, se construye una partición inicial de los individuos de forma aleatoria, la cual se identifica con un vector de tamaño n, donde n es el número total de individuos por clasificar. La $i-$ésima entrada de dicho vector contiene el número de clase al que pertenecerá el $i-$ésimo individuo.

Se entiende como una *partición candidata* a una partición que se obtiene al transferir un individuo x_i de una clase a otra. De esta manera, *un vecindario* de una partición P

corresponde al conjunto formado por todas las particiones candidatas de P, el cual se denota $N(P)$.

Una partición se considera tabú si ya ha sido generada en una cierta cantidad de iteraciones previas. Por su parte, la lista tabú está conformada por las últimas particiones generadas en el algoritmo y su tamaño es un parámetro por ajustar. Sin embargo, en Murillo (2000) se indica que lo que se pone en la lista tabú es una representación de la clase. De esta forma se evita el ciclado, pues al no aceptar una partición que contiene una clase prohibida de una partición ya analizada, no es posible llegar a una partición cuya vecindad ya fue estudiada.

Una partición de $N(P)$ es *admisible* si no es tabú o si el criterio de aspiración elimina su condición de tabú. Luego, si se denota con V^* al conjunto de todas las particiones admisibles de $N(P)$, entonces en cada iteración se escoge la mejor partición P' de V^* y la lista tabú se actualiza con la partición P, aunque la inercia intraclases de P' sea peor que la inercia intraclases de P. Con esto se busca trascender la optimalidad local.

En función de lo anterior, en dicho artículo se propone un algoritmo mejorado de búsqueda tabú, el cual se muestra en el algoritmo 4. Se puede notar que éste adolece de la aplicación de los procesos de intensificación y diversificación, los cuales, se indica en Murillo (2000), serían analizados posteriormente como mejoras potenciales al algoritmo.

De manera similar, Murillo & Trejos (1996) reportan cómo aplicaron la búsqueda tabú en el problema de particionamiento de datos, definiendo un vecindario de una partición por la transferencia de un objeto o individuo a una nueva clase. La lista tabú fue construida con indicadores para las clases de los objetos que fueron trasladados en las últimas m iteraciones. Dicha lista describe un conjunto de particiones que no pueden ser usadas a menos que sean las mejores particiones, en el sentido de la inercia intraclases, que hayan sido encontradas por el algoritmo. Los autores indican que el parámetro m debe

Algoritmo 4 Algoritmo de BT en Murillo (2000)

```
1: Determine la partición inicial aleatoria.
2: Mejor partición ← partición inicial.
3: Partición anterior ← partición inicial.
4: k = 0 // Contador de iteraciones.
5: mientras k < NumMaxIteraciones hacer
6:    k = k + 1.
7:    Cree el vecindario de la partición actual.
8:    Modifique la inercia intraclases (usando ΔW) de cada uno del vecindario.
9:    Escoja la mejor partición admisible.
10:   Actualice la lista tabú con la clase que contiene al individuo transferido.
11:   Modifique los centros de gravedad de la mejor partición admisible.
12:   Partición anterior ← mejor partición admisible.
13:   si Inercia mejor partición admisible < Inercia mejor partición entonces
14:      Mejor partición ← mejor partición admisible
15:   fin si
16: fin mientras
17: retornar Mejor partición encontrada.
```

ser ajustado y que el cálculo de ΔW (cambio en la inercia total) mediante las fórmulas de actualización, redujo considerablemente el tiempo de ejecución del algoritmo (Murillo & Trejos, 1996) .

Como se ha mostrado, a pesar de que el tema de las heurísticas en problemas de optimización combinatoria ha sido abordado con cierto detalle en las últimas dos décadas y, en algunos casos, con énfasis al problema de particionamiento con datos, igual sigue siendo un problema abierto y con muchas aristas por explorar. Los algoritmos heurísticos tienen la particularidad de que a pesar de que responden a una estructura base (un algoritmo clásico, con ciertas características inherentes e ineludibles del algoritmo), permiten mucha flexibilidad de implementación computacional. Esta característica, en particular, permite la búsqueda de implementaciones cada vez más eficientes y con un mejor rendimiento computacional.

Por otra parte, los algoritmos heurísticos tienen la característica adicional de depender

de muchos parámetros que deben ser ajustados en la implementación, los cuales pueden influir considerablemente en la eficiencia de los algoritmos. El ajuste de parámetros no es una tarea sencilla y existe una necesidad de no solo comparar el rendimiento entre heurísticas para estudiar el problema de particionamiento de datos, sino también realizar una calibración de los parámetros de cada heurística, de tal manera que se pueda explotar a plenitud su potencial para estudiar el problema en cuestión.

En esa línea, en Trejos et al. (1998) se indica la necesidad de un análisis más profundo de los parámetros en los tres métodos que ellos estudiaron: la longitud de las cadenas de Markov, el decrecimiento del parámetro de control, así como el parámetro inicial y final en el algoritmo de sobrecalentamiento simulado. Por su parte, el criterio de paro y las probabilidades asociadas al algoritmo genético. Y finalmente, el tamaño de la lista tabú y el número máximo de iteraciones en el algoritmo de búsqueda tabú.

1.5. Objetivos del proyecto

Habiendo estudiado con detalle el problema de optimización y sus principales antecedentes, a continuación se presentan el objetivo general y los objetivos específicos que enmarcaron el proceso de investigación.

Objetivo general

> Implementar heurísticas de optimización combinatoria para estudiar el problema de la clasificación por particiones en presencia de datos cuantitativos y el diseño e implementación de una nueva heurística híbrida de optimización para el estudio de dicho problema.

Dicho objetivo se alcanzó acorde con los siguientes objetivos específicos.

Objetivos específicos

1. Implementar las siguientes tres heurísticas para abordar el problema de clasificación por particiones en presencia de datos cuantitativos: algoritmo genético, enjambres de partículas y búsqueda tabú.

2. Explorar la variabilidad de los resultados en cada una de las heurísticas implementadas en función de sus parámetros.

3. Comparar la eficiencia de las heurísticas implementadas.

4. Diseñar e implementar un algoritmo híbrido que permita mejorar las soluciones.

1.6. Recursos empleados en el proyecto

Los algoritmos fueron programados en el software Borland Delphi Enterprise, Lite Edition, v7.3.4.2. Este software fue escogido por la fortaleza que tiene para el procesamiento eficiente de datos. Además, este lenguaje es orientado a objetos, lo cual responde a la metodología de programación seguida en el proyecto.

Con el objetivo de controlar la implementación de los algoritmos, se utilizaron tablas de datos cuantitativos diseñadas para tal fin. Además de tablas clásicas de la literatura del Análisis de Datos (*French Scholar Notes, Amiard's Fishes, the Thomas's Sociomatrix* y *Fisher's Iris*), se utilizaron las 16 tablas de datos simulados elaboradas por la M.Sc. Alexia Pacheco, que fueron empleadas en Trejos et al. (2006) para realizar una comparación de las metaheurísticas sobrecalentamiento simulado, búsqueda tabú, algoritmos genéticos, colonias de hormigas, enjambres de partículas y los métodos de k-medias y el de Ward, en el tema de particionamiento de datos cuantitativos. Estos resultados se usaron como base

de comparación y, además, se diseñaron ocho tablas adicionales con variables aleatorias normales con las siguientes características:

1. Las ocho tablas fueron diseñadas con el objetivo de estudiar el problema de particionamiento en siete clases.
2. En el diseño de las tablas se consideraron cuatro variantes: que las siete clases tuvieran la misma cardinalidad o no, y que las siete clases tuvieran la misma desviación estándar o no.
3. Cuatro de las tablas son de 1050 individuos y las cuatro restantes son de 2100 individuos, toda ellas consideradas en seis variables cuantitativas independientes.

Capítulo 2

Apartado 9: Metodología

En este capítulo se describe la metodología seguida en la implementación de los algoritmos. Por la naturaleza del proyecto de diseño e implementación de algoritmos, este capítulo se enfocará en la descripción de las estrategias técnicas seguidas en las diferentes implementaciones.

2.1. Algoritmo genético aplicado a particionamiento

La implementación del algoritmo genético para el problema de particionamiento en presencia de datos cuantitativos, está fundamentada principalmente en la estructura base que se presentó en el seudocódigo del algoritmo 1. Sin embargo, el método de paro y una mejora de centros móviles o $k-$medias fueron implementados para incrementar el rendimiento del programa. Una estructura más detallada de la implementación de este algoritmo genético se presenta en el algoritmo 5.

Es importante, para una mayor claridad del seudocódigo presentado en el algoritmo 5, realizar una descripción más profunda de los principales procesos y procedimientos que

se realizan en el mismo, específicamente se comentarán procesos como construcción de la población inicial de instancias, el cruce, la selección y la mutación, así como el método de paro y la mejora de $k-$medias que fue implementada para incrementar la eficiencia del algoritmo.

Algoritmo 5 Algoritmo genético implementado

Entrada: MaxIteraSinMejora, AplicarKMediasCada, m.
Salida: El individuo mejor adaptado (mejor solución obtenida).

1: Generar una población inicial Ψ, con m individuos.
2: Calcular el fitness de cada individuo en Ψ.
3: Contador $\leftarrow 0$.
4: ContadorIteraSinMejora $\leftarrow 0$.
5: MejorInerciaDeLaHistoria $\leftarrow \infty$.
6: **mientras** ContadorIteraSinMejora $<$ MaxIteraSinMejora **hacer**
7: Contador $\leftarrow$ Contador $+1$.
8: ContadorIteraSinMejora $\leftarrow$ ContadorIteraSinMejora $+1$.
9: $\Psi \leftarrow$ Cruces(Ψ).
10: **si** Contador **mod** AplicarKMediasCada $=0$ **entonces**
11: Se aplica una iteración completa de $k-$medias a cada una de las instancias que conforman la población Ψ.
12: **fin si**
13: $\Psi \leftarrow$ Mutación(Ψ).
14: $\Psi \leftarrow$ Selección(Ψ).
15: **si** alguna instancia mejoran la MejorInerciaDeLaHistoria **entonces**
16: MejorInerciaDeLaHistoria $\leftarrow$ Mejor Inercia en Ψ.
17: ContadorIteraSinMejora $\leftarrow 0$.
18: **fin si**
19: **fin mientras**
20: Para la mejor solución obtenida luego del proceso es importante determinar los centros de gravedad reales con el objetivo de determinar la inercia real.
21: **retornar** $I_0 \in \Psi$, donde I_0 es la mejor solución obtenida.

Para realizar dicha descripción primero se concretizarán las estructuras básicas empleadas en el proceso así como las principales variables que interactúan durante la ejecución del método implementado.

Instancia: Es uno de los miembros de la población que evolucionará a lo largo del al-

goritmo. Cada instancia debe manejar cierta cantidad de información tales como la codificación cromosómica de la solución que representa, el fitness que será el recíproco de la inercia intra-clase de la clasificación actual y los centroides que son los centros de gravedad para la clasificación actual.

La representación cromosómica de cada una de las instancias que define su código genético se puede representar como un vector fila de tamaño n tal que la $i-$ésima entrada contiene el índice de la clase a la que pertenece el $i-$ésimo individuo $\mathbf{x}_i$, tal como se presenta en las tablas 2.1 y 2.2.

Tabla 2.1: Representación de código genético de una instancia.

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\cdots$	$\mathbf{x}_i$	$\cdots$	$\mathbf{x}_n$
VClasificacion			$\cdots$		$\cdots$	

Tabla 2.2: Representación de código genético de una instancia para $n = 10$ y $k = 4$.

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$
VClasificacion	2	3	4	4	1	2	1	4	2	4

La tabla 2.2 indica que la partición de los datos es $P = \{C_1, C_2, C_3, C_4\}$ donde:

$$\begin{aligned} C_1 &= \{\mathbf{x}_5, \mathbf{x}_7\} \\ C_2 &= \{\mathbf{x}_1, \mathbf{x}_6, \mathbf{x}_9\} \\ C_3 &= \{\mathbf{x}_2\} \\ C_4 &= \{\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_8, \mathbf{x}_{10}\}. \end{aligned}$$

Población: Es concebida como un vector cuyas entradas son instancias. La población representa al conjunto de instancias que conforman a la generación actual en un tiempo dado. El modelo poblacional utilizado para tal fin fue el modelo de estado

estacionario mencionado en Abbass et al. (2002), ya que en el proceso de selección, algunos individuos de la población actual podrían ser seleccionados para formar parte de la nueva generación.

El tamaño de la población (número de instancias contenidas en la población) es un parámetro m. Durante la ejecución del programa el número de instancias dentro de la población aumentará durante el proceso de cruce y regresará a m instancias durante el proceso de selección.

Individuos a clasificar: Si cada uno de los objetos a clasificar está conformado por la observación de p variables cuantitativas, entonces cada uno de éstos puede ser representado como un vector en $\mathbb{R}^p$ de la forma:

$$\mathbf{x}_i = (x_{i1}, x_{i2}, x_{i3}, \cdots, x_{ip}).$$

De esta manera, si se desea clasificar n individuos: $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_n$, cada uno descrito por p variables $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \ldots, \mathbf{v}_p$, éstos se pueden representar como una tabla de n filas y p columnas, tal como la presentada en la tabla 2.3.

Tabla 2.3: Representación de los individuos a clasificar.

Individuos	$\mathbf{v}_1$	$\mathbf{v}_2$	$\mathbf{v}_3$	$\ldots$	$\mathbf{v}_p$
$\mathbf{x}_1$	x_{11}	x_{12}	x_{13}	$\ldots$	x_{1p}
$\mathbf{x}_2$	x_{21}	x_{22}	x_{23}	$\ldots$	x_{2p}
$\mathbf{x}_3$	x_{31}	x_{32}	x_{33}	$\ldots$	x_{3p}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\mathbf{x}_n$	x_{n1}	x_{n2}	x_{n3}	$\ldots$	x_{np}

Número de clases: Se denota con la variable K y representa la cantidad de grupos que se desean formar durante el proceso de optimización. Este parámetro es ingresado

por el usuario.

Probabilidad de mutar y cruzar: Son dos parámetros de entrada denotados por p_m y p_c respectivamente, al ser probabilidades, estos parámetros de ajuste cumplen $0 \leq p_m, p_c \leq 1$. Estos mismos se utilizan en el proceso de mutación y cruce presentes en el algoritmo.

2.1.1. Descripción del algoritmo genético

Suponga que se desea clasificar n individuos descritos por p variables en K clases distintas, de manera que ninguna de ellas quede vacía. Para ello se empleará un total de m instancias en la primera generación. A continuación se detallarán los principales aspectos presentes en el algoritmo 5.

Construcción de la población inicial de instancias. Este proceso se realiza solo una vez en la ejecución del algoritmo. Se escogió realizar la construcción de la población inicial de instancias de forma aleatoria, pues es el más común y ha mostrado buenos resultados según de los Cobos et al. (2010); Mitchell (1999); Talbi (2009). La generación no aleatoria puede generar convergencia prematura, al respecto Moujahid et al. (sf) indica que la generación no aleatoria de las instancias de la población inicial podría acelerar la convergencia del algoritmo, teniendo el inconveniente de que esta aceleración podría generar que dicha convergencia se realice a óptimos locales. De esta manera, para esta implementación las m instancias que conforman la población inicial, fueron generadas al azar en el espacio de búsqueda de la siguiente forma:

- Se determinan los valores extremos para cada una de las p variables de la tabla de datos original (ver tabla 2.3), de manera que se pueda calcular el conjunto D que corresponde al menor hiperrectángulo de $\mathbb{R}^p$ que contiene a todos los n

individuos a clasificar. Es decir, si $\mathbf{y} = (y_1, y_2, \ldots, y_p) \in D$, entonces:

$$\text{mín}\{x_{ij} : i = 1, 2, \ldots, n\} \leq y_j \leq \text{máx}\{x_{ij} : i = 1, 2, \ldots, n\},$$

para todo $j = 1, 2, \ldots, p$.

- Ahora se deben fabricar las m instancias de la población inicial, cada una de ellas se construye por medio del siguiente procedimiento.
  - Se generan, aleatoriamente, K puntos $\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \ldots, \mathbf{g}_K$ en D de la forma:

    $$\mathbf{g}_r = (g_1^r, g_2^r, g_3^r, \ldots, g_p^r),$$

    donde K es el número de clases a formar. Estos puntos funcionarán como centros de gravedad artificiales (centroides) para realizar una primera clasificación. El punto $\mathbf{g}_r \in D$ será asignado como centroide de la clases C_r a formar en la primera clasificación, con $r = 1, 2, \ldots, K$.
  - Cada uno de los individuos a clasificar es asignado a la clase cuyo centro de gravedad artificial esté más cercano. Es decir, para todo $i = 1, 2, 3, \ldots, n$ el individuo $\mathbf{x}_i$ es asignado la clase C_l si

    $$d_{il} = \|\mathbf{x}_i - \mathbf{g}_l\|^2 = \text{mín}\{\|\mathbf{x}_i - \mathbf{g}_r\|^2, \text{ con } r = 1, 2, 3, \ldots, K\}.$$

    Este proceso realiza una primera clasificación que será considerado el código genético de la instancia generada.

Operador de cruce: El objetivo principal del operador de cruce es explorar la mayor parte de la información contenida en la población por medio de la generación de nuevas soluciones que heredan algunas características de dos padres.

Cuando la población entra al proceso de cruce, para cada una de las instancias en

la población se verá, bajo una probabilidad, si será cruzada o no. Esto se logra generando, para cada una de ellas, un número aleatorio entre cero y la unidad, si este número aleatorio es menor que p_c, entonces la instancia será cruzada.

La pareja del cruce será seleccionada al azar del total de instancias en la población, teniendo cuidado de no cruzar una instancia con sí misma. De esta manera, cada una de las instancias puede ser cruzada por una de dos razones, fue seleccionada por la probabilidad, o fue seleccionada como pareja por otra instancia de la población.

En Gil (2006) se presentan diferentes procedimientos de cruce, mientras que por otro lado Moujahid et al. (sf) investigó sobre los método de cruce basados en puntos de corte, este autor indica que si bien el cruce basado en dos puntos representa una mejora en el algoritmo genético, la utilización de tres o más puntos no beneficia el comportamiento del mismo. En Talbi (2009) se indica que el diseño del operador de cruces depende de la codificación y de la naturaleza del problema a resolver, este autor establece dos consideraciones que se deben tener en cuenta durante el diseño del operador: la heredabilidad de características y la validez de las soluciones generadas por el cruce.

Por la singularidad del problema a resolver, se diseñó un operador de cruce cuya principal particularidad es que hereda la mejor característica del padre dominante, donde el padre dominante es aquel que tiene mejor inercia intra-clase (inercia intra-clase más baja), mientras que al otro se le denominará padre recesivo.

En este operador de cruce, el padre dominante heredará la clase con mayor concurrencia, mientras que las restantes entradas del código genético serán tomadas del padre recesivo. La razón de que se considere buena idea que el padre de mejor inercia intra-clase herede la clase más grande, proviene del hecho que una clase grande en una instancia con baja inercia es considerada una buena clase, puesto que los individuos en ella deben estar muy juntos entre sí.

Para entender mejor este operador, considere el siguiente ejemplo:

Suponga que se desea clasificar 10 individuos en 3 clases. Suponga además, que se tiene dos instancias I_1 e I_2, tales que I_1 tiene menos inercia que I_2. Sean

$$\begin{array}{rcl} \mathsf{VClasificacion}_1 & = & \boxed{1}\boxed{2}\boxed{1}\boxed{3}\boxed{2}\boxed{2}\boxed{2}\boxed{1}\boxed{2}\boxed{3} \\ \mathsf{VClasificacion}_2 & = & \boxed{3}\boxed{1}\boxed{3}\boxed{1}\boxed{2}\boxed{3}\boxed{3}\boxed{3}\boxed{1}\boxed{2}\,, \end{array}$$

el código genético de I_1 e I_2, respectivamente.

Como I_1 tiene mejor inercia que I_2, entonces I_1 es el padre dominante mientras que I_2 es el padre recesivo. Note que la clase de mayor concurrencia en I_1 es la clase 2 con cinco individuos. De esta manera, la instancia I_1 heredará la clase 2 al hijo H mientras que I_2 heredará las restantes entradas para completar el código genético del hijo H.

$$\begin{array}{rcl} \mathsf{VClasificacion}_1 & = & \boxed{1}\boxed{2}\boxed{1}\boxed{3}\boxed{2}\boxed{2}\boxed{2}\boxed{1}\boxed{2}\boxed{3} \\ & & \quad\downarrow\qquad\quad\downarrow\;\downarrow\;\downarrow\qquad\downarrow \\ \mathsf{VClasificacion}_H & = & \boxed{3}\boxed{2}\boxed{3}\boxed{1}\boxed{2}\boxed{2}\boxed{2}\boxed{3}\boxed{2}\boxed{2} \\ & & \uparrow\qquad\uparrow\;\uparrow\qquad\qquad\uparrow\qquad\uparrow \\ \mathsf{VClasificacion}_2 & = & \boxed{3}\boxed{1}\boxed{3}\boxed{1}\boxed{2}\boxed{3}\boxed{3}\boxed{3}\boxed{1}\boxed{2}\,. \end{array}$$

De este modo, se tiene que la instancia H tendrá como código genético a:

$$\mathsf{VClasificacion}_H = \boxed{3}\boxed{2}\boxed{3}\boxed{1}\boxed{2}\boxed{2}\boxed{2}\boxed{3}\boxed{2}\boxed{2}\,.$$

Finalmente, para completar la construcción del hijo H, es necesario determinar los centroides de cada una de las clases y la inercia intra-clase. Esta información deberá ser calculada respetando la clasificación presente en el código genético del hijo.

Operador de mutación: El operador de mutación se encarga de realizar cambios aleatorios en parte del código genético (generalmente un alelo) de algunas instancias de la población, con el objetivo de garantizar siempre suficientes variaciones dentro de la población. Esta variación se realiza sobre los individuos de la población que sean escogidos bajo una probabilidad p_m. Es decir, cada individuos en la población tiene un p_m de probabilidad de ser mutado.

Según Lee & El-Sharkawi (2008) en los últimos años se ha presentado un creciente reconocimiento de la importancia de la mutación viéndolo como el responsable de la reintroducción de genes y valores perdidos accidentalmente. Este mismo autor asegura que el operador de mutación es más importante en las generaciones finales, cuando la mayoría de los individuos presentan una calidad similar y sus códigos genéticos podrían ser más parecidos, evitando así la convergencia prematura del algoritmo y dando mayor tiempo de exploración del espacio de búsqueda; más aún, se muta para garantizar que ninguna región del espacio de búsqueda tenga probabilidad nula de ser explorada (Gestal, sf) , pues cualquier secuencia, en el código de los cromosomas, puede ser construida con un número finito de mutaciones.

Por la particularidad del problema y del código genético de cada instancia se diseñó el operador de mutación basado en la transferencia de un individuo a una nueva clase, este operador de mutación fue implementado por Piza et al. (1999) produciendo buenos resultados.

Cuando la población de instancias entra en el proceso de mutación, se debe analizar, bajo una probabilidad p_m, si cada una de las instancias de la población serán mutada o no. Esto se logra generando, para cada instancia, un número aleatorio entre cero y

la unidad, si este número es menor que p_m, entonces la instancia será mutada; y el resultado de dicha mutación remplazará la instancia original, no aumentando así el tamaño de la población.

El proceso de mutación de una instancia consiste en seleccionar una posición aleatoria del código genético (un alelo) y cambiar el valor actual por un valor entre 1 y K generado en forma aleatoria, evitando colocar el mismo valor que traía originalmente. Este procedimiento es equivalente a cambiar un individuo, seleccionado al azar, a una nueva clase seleccionada también al azar.

Finalmente, para terminar el proceso de mutación de una instancia en particular, es necesario actualizar el valor de la inercia intra-clase, así como los centroides de las clases involucradas. Para esto se utilizó las fórmulas de actualización presentadas en la sección **??** de este reporte.

Operador de selección: Este operador tiene la tarea de seleccionar a los individuos que conformarán la nueva generación de instancias. Aunque existen diversos método de selección, mismos que pueden ser consultados en Gestal (sf); Gil (2006); Lee & El-Sharkawi (2008), el *Stochastic Universal Sampling* (SUS) presentado en Talbi (2009) es simple de implementar, de bajo costo computacional y ha demostrado buenos resultados pues garantiza cierta variabilidad en la escogencia de las instancias. Por esta razón el mismo fue escogido para ser implementado en el algoritmo propuesto. El SUS se expone con detalle a continuación.

Suponga que se desean seleccionar, de un conjunto mayor, un total de m instancias para conformar la nueva generación. El método SUS, consiste en construir una ruleta circular con todas las instancias de la población, de manera que el arco de la circunferencia que le corresponde a cada instancia sea proporcional al valor de ajuste (o en este caso inversamente proporcional a la inercia intra-clase). Luego se colocan m marcadores igualmente espaciados a lo largo de la circunferencia de la

ruleta. De esta manera, con un solo giro de la ruleta, es posible seleccionar a todas las instancias al mismo tiempo.

Para comprender mejor el método, considere el siguiente ejemplo. Suponga que se desean escoger 5 instancias del conjunto:

$$\Psi = \{I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9\}$$

cuyas inercia intra-clase y valor de aptitud se presentan en la tabla siguiente:

Tabla 2.4: Tabla de inercia intra-clase y valor de aptitud para cada instancia.

Instancias	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9
Inercia intra-clase $W(I_i)$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{2}$	1	1	1	$\frac{1}{7}$
Valor de aptitud	2	2	3	5	2	1	1	1	7

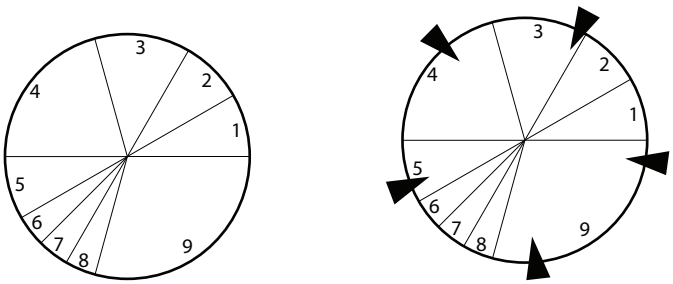
La construcción de la ruleta para el método SUS (ver figura 2.1) se realiza de la siguiente manera:

- Considere una ruleta circular. Como la suma del los valores de aptitud es 24, entonces se deberá dividir la circunferencia de la ruleta en 24 arcos de igual longitud.
- De los datos presentes en la tabla 2.4, se seleccionará el mismo número de arcos como indica su valor de aptitud para cada uno de las instancias presentes en la población. Obteniendo la ruleta de la figura 2.1(a).
- Se coloca, para este caso particular, cinco marcadores igualmente distribuidos a lo largo de la circunferencia de la ruleta.

De esta manera, haciendo girar la ruleta una sola vez se pueden seleccionar las cinco instancias que se necesitan. Si luego de hacer girar dicha ruleta, el resultado

es el que se presenta en la figura 2.1(b), entonces la nueva generación de instancias estará conformada como sigue:

$$\Psi = \{I_3, I_4, I_5, I_9, I_9\}$$



(a) Ruleta de método SUS.

(b) Marcadores igualmente distribuidos.

Figura 2.1: Construcción de la ruleta para el método SUS.

Note que el método de selección SUS puede seleccionar dos o más veces una instancia, tal como ocurrió con la instancia I_9, además las instancias con mejor valor de aptitud tiene mayor probabilidad de ser seleccionadas, más aún, si el valor de aptitud de una instancia es mayor que la distancia entre los marcadores de la ruleta, ésta tiene probabilidad 1 de ser seleccionada, manteniendo así un componente de elitismo en la selección. Mientras que las instancias con fitness desfavorables tienen menos oportunidad de ser seleccionadas y algunas no tiene posibilidad de ser seleccionadas más de una vez. La ruleta simple, por su parte, puede seleccionar muchas veces instancias con bajo fitness y no seleccionar instancias con fitness altos, dado que la escogencia de cada una es independiente de la escogencia de otra, (ver *selección por sorteo o ruleta simple* en la página 30).

Para el algoritmo implementado siempre se tendrá el cuidado de seleccionar, forzadamente, la instancia que tenga mejor inercia. Luego las restantes $m - 1$ instancias

serán seleccionadas por medio el método SUS.

Mejora de $k-$medias: Esta mejora cosiste en aplicar el método de $k-$medias después de un determinado número de iteraciones del algoritmo, esta idea fue presentada por Trejos et al. (2006) y Piza et al. (1999) quienes combinaron los operadores genéticos con el método de nubes dinámicas o centros móviles ($k-$medias). Estos autores reportan mejores resultados si después de cierto número de iteraciones, cada una de las particiones formadas se hacen converger hacia un óptimo local por medio de la aplicación del método de Forgy.

Un parámetro de entrada AplicarKMediasCada es ajustado por el usuario y se encarga de indicarle al algoritmo, después de cuantas interaciones debe aplicar la mejora propuesta. Esta mejora, se realiza cuando el número de iteraciones realizadas es divisible entre el valor del parámetro AplicarKMediasCada.

Cuando la población completa inicia la aplicación del operador de $k-$medias, a cada una de las instancias de la población se somete al procedimiento que se presenta en el algoritmo 6.

Note que el procedimiento no se ejecuta hasta que se garantice la convergencia, sino hasta que la diferencia entre la inercia anterior y la actual, no supere el umbral de 0.001. En algunas pruebas previas con tablas de control no fue posible detectar diferencias significativas en el rendimiento de la heurística para umbrales más reducidos. Los resultados de estas pruebas previas pueden consultarse en la tabla 2.5 para los datos Tabla-105-K(7)Card($\neq$)DS($\neq$) y Tabla-525-K(7)Card($\neq$)DS($\neq$).

Algoritmo 6 Mejora de $k-$medias

Entrada: Instancia de entrada I.

Salida: Instancias luego de aplicar $k-$medias.

1: InerciaAnterior$\leftarrow -1$.
2: **mientras** $|\text{InerciaAnterior} - W(I)| > 0.001$ **hacer**
3: InerciaAnterior$\leftarrow W(I)$.
4: Se recalculan los centroides de I para la clasificación de la instancia.
5: Dado los centroides de I determinado en el paso anterior, se construye una nueva clasificación asignado a cada uno de los individuos a clasificar el centroide más cercano.
6: Se actualiza $W(I)$.
7: **fin mientras**
8: **retornar** La instancia resultante.

Tabla 2.5: Porcentajes de atracción y tiempos promedios de ejecución en segundos para diferentes umbrales en la convergencia de $k-$medias.

	Umbral 0.001		**Umbral 0.00001**		**Umbral 0.0000001**	
	% de atracción	Tiempo promedio	% de atracción	Tiempo promedio	% de atracción	Tiempo promedio
Tabla-105	48.20 %	0.032 292s	48.40 %	0.032 974s	47.50 %	0.031 924s
Tabla-525	34.80 %	0.154 995s	34.70 %	0.158 180s	35.00 %	0.157 389s

Método de paro del algoritmo: Generalmente, los criterio de paro que estudian algunos indicadores del estado evolutivo de la población de instancias son, por lo general, computacionalmente costoso, por lo que su uso tiene una incidencia directa en el tiempo de ejecución del algoritmo. Además, a menos que se utiliza de forma combinada con otros, pueden ocasionar convergencias prematuras de la población de instancias. Es decir, los criterios que solo miden el estado evolutivo de la población podrían detener el algoritmo antes de tiempo evitando alcanzar el óptimo global del

problema a resolver.

Detener el algoritmo cuando haya pasado un cierto número de iteraciones sin que haya una mejora en la mejor inercia reportada hasta el momento, parece ser una buena opción para el método de paro. Este método no requiere de cálculos engorrosos y costosos, además tiene la ventaja de mitigar un poco el efecto de una convergencia prematura, dando tiempo de que el algoritmo escape de óptimos locales. Por esta razón este método citado por Gil (2006) fue una opción a considerar en la implementación del algoritmo.

2.2. Algoritmo basado en enjambres de partículas en particionamiento

La implementación del algoritmo basado en enjambres de partícula para el problema de particionamiento con datos cuantitativos está basado en el seudocódigo presentado en el algoritmo 2. Sin embargo, se han adicionado, a los operadores presentes propios del algoritmo, una mejora que consiste en la aplicación del método de Forgy ($k-$medias), mismo que significó una mejora sustancial en el Algoritmo Genéticos.

En el algoritmo 7 se presenta el seudocódigo específico de la implementación de la heurística.

Para un mejor entendimiento del método propuesto en el algortimo 7 es necesario realizar una descripción más detallada de los principales procedimientos y operadores que interactuan en la ejecución del mismo. Discutir procesos como la construcción del enjambre de partículas, la actualización de la velocidad, la forma de mover las partículas dentro del enjambre, la mejora de $k-$medias y el método de paro propuesto son algunos

Algoritmo 7 Algoritmo de optimización por enjambres de partículas

Entrada: Parámetros de entradas α, c_1, c_2.

Salida: La mejor posición encontrada por el enjambre.

1: Construye aleatoriamente el enjambre $\mathcal{E} = (\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m)$.
2: Contador $\leftarrow 0$.
3: **mientras** ContadorIteraSinMejora $<$ MaxIteraSinMejora **hacer**
4: Evaluar el costo de cada partícula: $W(\mathcal{P}_k)$, para $k = 1, 2, 3, \ldots m$
5: **para** $k \leftarrow 1$ **hasta** m **hacer**
6: $\kappa_1 \leftarrow c_1 \cdot \text{rnd}(0, 1)$
7: $\kappa_2 \leftarrow c_2 \cdot \text{rnd}(0, 1)$
8: Actualizar la velocidad de la $k-$ésima partícula $\overrightarrow{\mathbf{v}}_k^{(t+1)}$:
9: $t \leftarrow$ Contador

$$v_{kj}^{(t+1)} \leftarrow \alpha v_{kj}^{(t)} + \kappa_1[p_{kj}^{(t)} - x_{kj}^{(t)}] + \kappa_2[\overrightarrow{\chi}_{lider,j}^{(t)} - x_{kj}^{(t)}]$$

10: Mover la $k-$ésima partícula a su nueva posición $\overrightarrow{\chi}_k^{(t+1)}$.

$$\overrightarrow{\chi}_k^{(t+1)} \leftarrow \overrightarrow{\chi}_k^{(t)} + \overrightarrow{\mathbf{v}}_k^{(t+1)}$$

11: Actualizar la mejor posición de la $k-$ésima partícula en caso de mejorar la existente. Actualizar $\overrightarrow{\chi\mathbf{p}_k}$.
12: Actualizar la mejor posición encontrada por el enjambre en caso de mejorar la existente. Actualizar $\overrightarrow{\chi\mathbf{m}}$.
13: **fin para**
14: Contador $\leftarrow$ Contador $+ 1$.
15: **si** ContadorIteraSinMejora es divisible entre AplicarKMediasCada **entonces**
16: Aplique $k-$medias a cada una de las partículas del enjambre.
17: **fin si**
18: **fin mientras**
19: **retornar** La mejor solución encontrada durante la búsqueda.

de los tópicos que serán discutidos en la descripción del algoritmo. Sin embargo, previo a estas descripciones es necesario esclarecer lo que se entenderá por enjambre, por el espacio de búsqueda y por partícula.

Enjambre: El enjambre de partículas es un vector de tamaño m, de la forma:

$$\mathcal{E} = \Big(\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \ldots, \mathcal{P}_m\Big),$$

donde m es un parámetro de entrada que representa el número de partículas presentes en el enjambre durante la ejecución del algoritmo. Y $\mathcal{P}_k$ representa la $k-$ésima partícula en $\mathcal{E}$.

Espacio de búsqueda: este conjunto denotado por Ω está conformado por todos los puntos en $\mathbb{R}^{p\cdot K}$ de la forma $\overrightarrow{\chi} = (\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_K)$, donde $\mathbf{g}_r = (g_1^r, g_2^r, \ldots, g_p^r)$ tal que g_j^r se encuentre entre mín$\{x_{ij} : i = 1, 2, \ldots, n\}$ y el máx$\{x_{ij} : i = 1, 2, \ldots, n\}$, para todo $r = 1, 2, 3, \ldots, K$ y para todo $j = 1, 2, 3, \ldots, p$.

Es decir, si D es el menor hiperrectángulo que contiene a los n individuos a clasificar, entonces el espacio de búsqueda será $\Omega = D^K$.

Partícula: Una partícula $\mathcal{P}$ es un individuo del enjambre $\mathcal{E}$. Cada una de las partículas está definida por las coordenadas de su posición actual $\overrightarrow{\chi}$ en el espacio de búsqueda Ω y las coordenadas de la mejor posición $\overrightarrow{\chi}\mathbf{p}$ reportada por la partícula (memoria). Puesto que las coordenadas de la partícula definen, por construcción, a los centroides de las clases a formar, la solución al problema de particionamiento se logra asignando cada uno de los datos $\mathbf{x}_i$ a la clase cuyo centroide $\mathbf{g}_j$ minimice la distancia $\|\mathbf{x}_i - \mathbf{g}_r\|^2$ para $r = 1, 2, 3, \ldots, K$. Por esta razón, si dos partículas tiene la mismas coordenadas representarán la misma solución al problema, sin embargo, dentro de cada una de las partículas se debe manejar información tal como la mejor posición reportada por

la partícula y la velocidad con la cual la partícula llegó a su posición actual, esta información adicional hace a la partícula única dentro del enjambre independiente de la posición actual.

2.2.1. Descripción del algoritmo basado en enjambres de partículas

Suponga que se desea clasificar n individuos definidos por p variables cuantitativas en K clases distintas, de manera que ninguna de ellas quede vacía. Para la cual se considera un enjambre $\mathcal{E}$ con m partículas. A continuación se detalla los principales aspectos presentes en el algoritmo.

Construcción del enjambre: Tal y como se menciona en Alias et al. (2011); Eberhart & Kennedy (1995); Hvass & Chipperfield (2010); las partículas pueden ser construidas utilizando posiciones aleatorias del espacio de búsqueda Ω, las cuales deben representar una solución factible. Así, la construcción de una partícula $\mathcal{P}$ en el enjambre se hace como sigue.

- Se genera un punto al azar en Ω. Este punto será asignado tanto a $\overrightarrow{\chi}$ como a $\overrightarrow{\chi}\mathbf{p}$.
- Se calcula la clasificación asociada a dicha posición del espacio Ω.
- Se calcula la inercia intra-clase $W(\mathcal{P})$ asociada a la partícula.
- Finalmente, se selecciona el líder del enjambre mismo que corresponde a la partícula con menor inercia en el tiempo actual. Es importante guardar en la memoria del enjambre la posición del líder y la mejor inercia reportada por cualquier miembro del enjambre en cualquier tiempo.

Actualización de la velocidad y posición: En la actualización de la velocidad, se hace una modificación a la fórmula presentadas en Alias et al. (2011); Hassan et al. (2005); Hvass & Chipperfield (2010); Lima & Barán (2006); Trejos & Villalobos (2007); Xie et al. (2002b). Estos autores proponen como fórmula de actualización de la velocidad en la $k-$ésima partícula la presentada en la fórmula (2.1).

$$\overrightarrow{\mathbf{v}}_k^{(t+1)} \leftarrow \alpha\overrightarrow{\mathbf{v}}_k^{(t)} + \kappa_1[\overrightarrow{\chi\mathbf{p}}_k^{(t)} - \overrightarrow{\chi}_k^{(t)}] + \kappa_2[\overrightarrow{\chi\mathbf{m}}^{(t)} - \overrightarrow{\chi}_k^{(t)}]. \tag{2.1}$$

La modificación consiste en sustituir, en la fórmula (2.1), el vector $\overrightarrow{\chi}\mathbf{m}$ por $\overrightarrow{\chi}_{lider}^{(t)}$ donde $\overrightarrow{\chi}_{lider}^{(t)}$ denota la posición del líder del enjambre en la iteración t, donde el líder en la iteración t corresponde a la partícula en dicha iteración que tiene menor inercia actual en el enjambre. De esta manera, la fórmula de actualización de la velocidad para la $k-$ésima partícula corresponde a:

$$\overrightarrow{\mathbf{v}}_k^{(t+1)} \leftarrow \alpha\overrightarrow{\mathbf{v}}_k^{(t)} + \kappa_1[\overrightarrow{\chi\mathbf{p}}_k^{(t)} - \overrightarrow{\chi}_k^{(t)}] + \kappa_2[\overrightarrow{\chi}_{lider}^{(t)} - \overrightarrow{\chi}_k^{(t)}]. \tag{2.2}$$

La razón para proponer el cambio corresponde a que en pruebas preliminares el cambio parece favorecer levemente el porcentaje de atracción. Por ejemplo, para los datos de Tabla-105-K(7)Card($\neq$)DS($\neq$) y Tabla-525-K(7)Card($\neq$)DS($\neq$), así como para la tabla de datos Tabla-105-K(7)Card($=$)DS($\neq$) los porcentajes estuvieron levemente mejor usando $\overrightarrow{\chi}_{lider}^{(t)}$ que usando $\overrightarrow{\chi}\mathbf{m}$. Los resultados de estas pruebas preliminares se puede apreciar en la tabla 2.6. En el tiempo de ejecución no parece existir diferencia.

Tabla 2.6: Comparación entre el uso del líder actual y el líder global en la fórmula de actualización de la velocidad.

	Usando $\overrightarrow{\chi}_{lider}^{(t)}$		Usando $\overrightarrow{\chi}\mathbf{m}$	
	% de atracción	Tiempo promedio	% de atracción	Tiempo promedio
Tabla-105-K(7)Card(≠)DS(≠)	96.30 %	0.014 062s	94.50 %	0.014 190s
Tabla-525-K(7)Card(≠)DS(≠)	92.40 %	0.077 854s	89.20 %	0.076 696s
Tabla-105-K(7)Card(=)DS(≠)	65.60 %	0.015 125s	63.70 %	0.015 013s

Finalmente, la actualización de la posición de la partícula se realiza por medio de la fórmula:

$$\overrightarrow{\chi_k}^{(t+1)} \leftarrow \overrightarrow{\chi_k}^{(t)} + \overrightarrow{\mathbf{v}_k}^{(t+1)}. \tag{2.3}$$

Mejora de $k-$medias: la mejora de $k-$medias consisten en aplicar el método de Forgy una vez que haya transcurrido un cierto número de iteraciones sin haber reportado una mejora, esto constituye una diferencia importante con la aplicación del método de Forgy en el algoritmo genético, puesto que en éste la mejora se aplica cada cierto número de iteraciones.

Este método consiste en hacer tender cada solución existente en el enjambre a un óptimo local mediante la aplicación del método $k-$medias clásico. Esta convergencia no es completa, puesto que se itera hasta que la mejora de la inercia sea menor a una tolerancia previamente definida, que para este caso dicha tolerancia se fijó en 0.01, mostrando generar buenos resultados. Similarmente a lo ocurrido en el caso de algoritmos genéticos, disminuir el umbral entre las inercias no parece mejorar el rendimiento del algoritmo, los detalles pueden ser consultados en la tabla 2.7.

Tabla 2.7: Porcentajes de atracción y tiempos promedios de ejecución en segundos para diferentes umbrales en la convergencia de $k-$medias.

	Umbral 0.01		Umbral 0.00001		Umbral 0.0000001	
	% de atracción	Tiempo promedio	% de atracción	Tiempo promedio	% de atracción	Tiempo promedio
Tabla-105	81.90 %	0.015 825s	81.80 %	0.016 272s	81.60 %	0.016 010s
Tabla-525	88.10 %	0.085 623s	88.70 %	0.102 416s	88.30 %	0.103 935s

De manera general, el método de $k-$medias aplicada a una partícula $\mathcal{P}$ se realiza tal y como se presenta en el algoritmo 8.

Algoritmo 8 Mejora de $k-$medias en enjambres de partículas

Entrada: Partícula de entrada $\mathcal{P}$.

Salida: Partícula luego de aplicar $k-$medias.

1: InerciaAnterior$\leftarrow -1$.
2: **mientras** $|\text{InerciaAnterior} - W(\mathcal{P})| > 0.01$ **hacer**
3: InerciaAnterior$\leftarrow W(\mathcal{P})$
4: Se determina la clasificación dado los centroides en el vector de coordenadas $\overrightarrow{\chi}_i$.
5: Se determina los centroides de cada una de las clases dada la clasificación calculada en el paso anterior.
6: Se actualiza $W(\mathcal{P})$.
7: **fin mientras**
8: **retornar** $\mathcal{P}$.

Método de paro del algoritmo: el método de paro propuesto por Hassan et al. (2005) y que corresponde al empleado para algoritmos genéticos donde demostró tener un buen desempeño, fue el implementado en la presente heurística. Este criterio de paro consiste en detener el algoritmo cuando haya transcurrido un número de iteraciones

predefinido sin reportar una mejora real en la mejor inercia determinada por el enjambre.

2.3. Búsqueda tabú aplicado a particionamiento

2.3.1. Preliminares

Se retoma el hecho que en el problema de particionamiento tratado se tiene como insumo la tabla de datos 1.1 de tamaño $n \times p$ (n individuos y p variables cuantitativas independientes). En dicha tabla, x_i es el $i-$ésimo individuo que corresponde a un vector en $\mathbb{R}^p$. Se quiere determinar una partición en K clases $P = \{C_1, \ldots C_K\}$ del conjunto $X = \{x_1, \ldots, x_n\}$, de tal manera que se minimice la inercia intraclases asociada a P, dada por:

$$W(P) = \frac{1}{n} \sum_{l=1}^{K} \sum_{x_i \in C_l} \|x_i - g_l\|^2 ,$$

donde g_l denota el centro de gravedad de la clase C_l y la norma usada denota la distancia euclídea clásica.

Conociendo lo anterior, para efectos de la implementación computacional se construye un vector, denominado VClasificacion (vector de clasificación), que representa la forma en la que se manejan las posibles particiones de X. Dicho vector posee entradas enteras y es de dimensión $1 \times n$. Además, tiene la forma:

$$\mathsf{VClasificacion} = (c_1, \ldots, c_n) ,$$

de tal manera que la $i-$ésima entrada, c_i, de VClasificacion satisface que $c_i \in \{1, \ldots, K\}$ y denota la clase a la que se asignará el individuo x_i de X. Por lo tanto, se quiere determinar

la combinación de entradas enteras para VClasificacion que minimice el valor de la inercia intraclases W y que represente de manera computacional la partición P buscada. Por lo anterior, en la implementación se entiende como una solución factible del problema a cada combinación posible tomada por VClasificacion. Las heurísticas de búsqueda tabú parten de una solución inicial generada de manera aleatoria. Esto es, una combinación dada sobre VClasificacion, en la que cada entrada de dicho vector es seleccionada de manera aleatoria del conjunto $\{1, \ldots, K\}$.

2.3.2. Generalidades de la implementación de los algoritmos de BT

En esta sección se describen las principales generalidades que forman parte de los algoritmos de búsqueda tabú. En particular, se explicará sobre la estrategia de diversificación y la aplicación del algoritmo de k-medias.

Como estrategia de diversificación, acorde con la alternativa expuesta en Lee & El-Sharkawi (2008) y Riojas (2005), se implementó la idea de reiniciar periódicamente la búsqueda desde puntos elegidos aleatoriamente del espacio de soluciones factibles. Adicionalmente, cada vez que se realiza la diversificación, la lista tabú correspondiente también es reiniciada. Para efectos del diseño del algoritmo y en específico para el control de la diversificación, se incorporó el parámetro ReiniciarSolActualCada, que controla cada cuántas iteraciones en las que el algoritmo no ha mejorado la solución de mayor calidad almacenada en memoria, se debe reiniciar la solución actual. Sobre la naturaleza y las características de la lista tabú se detallará posteriormente, cuando se aborden las especificidades propias de los algoritmos implementados.

Por otra parte, en las dos versiones del método de búsqueda tabú se utilizó el algoritmo

de k-medias. En ambos casos, durante la construcción del vecindario de una solución factible[1] se sigue el proceso de generar un vecino e inmediatamente aplicarle el algoritmo de k-medias (ver línea 5 en el algoritmo 10). Esta estrategia mejoró considerablemente la calidad, en términos de la convergencia de las heurísticas al óptimo global.

El criterio de aspiración fue aplicado como estrategia para aceptar un vecino de una solución que tiene la particularidad de tener la condición tabú, siempre que la inercia intraclases asociada a dicho vecino, sea menor que la inercia intraclases de la mejor solución obtenida hasta el momento. La implementación de dicho criterio puede observarse en la línea 12 del algoritmo 10.

Es importante hacer énfasis en que se hicieron pruebas para aplicar una estrategia de intensificación en ambos algoritmos. Dicho proceso se implementó en el sentido expuesto en Lee & El-Sharkawi (2008), consistiendo en explorar vecindarios de soluciones élites previamente encontradas en el proceso. Esta aplicación no generó una mejora significativa en la ejecución de los algoritmos, comparando con los resultados que ya se tenían antes de hacer la intensificación. Por este motivo, se decidió reportar los resultados de los algoritmos de búsqueda tabú sin una estrategia de intensificación. Para contextualizar dichas pruebas, en las tablas 2.8 y 2.9 se muestran los resultados obtenidos al aplicar búsqueda tabú en las tablas A12, A15 y A16. En todos los casos, se ejecutaron los algoritmos con los parámetros $\mathsf{CantidadVecinos} = 4$ y $\mathsf{ReiniciarSolActualCada} = 5$. Los porcentajes de atracción se muestran en la columna encabezada con el símbolo "%".

Finalmente, con respecto al método de parada se utilizó el criterio de detener los algoritmos después de haber transcurrido un número máximo de iteraciones.

[1] En la práctica no se construye el vecindario completo pues normalmente éste está compuesto por demasiadas soluciones factibles. Por lo tanto, en la implementación se elige aleatoriamente una cierta cantidad de dichas soluciones. Esa cantidad es un parámetro para los algoritmos de BT y fue denotado con CantidadVecinos.

Tabla 2.8: Resultados obtenidos en la heurística de BT por transferencias, en pruebas preliminares realizadas para analizar si se aplicaba o no un proceso de intensificación.

		Sin intensificación		**Con intensificación**	
Tabla	**Iteraciones**	**%**	**Tiempo(s)**	**%**	**Tiempo(s)**
A12	120	79 %	0.341753	77 %	0.366401
A15	50	76 %	0.026359	80 %	0.023783
A16	120	75 %	0.337188	77 %	0.352775

Tabla 2.9: Resultados obtenidos en la heurística de BT mediante movimiento de centros de gravedad, en pruebas preliminares realizadas para analizar si se aplicaba o no un proceso de intensificación.

		Sin intensificación		**Con intensificación**	
Tabla	**Iteraciones**	**%**	**Tiempo(s)**	**%**	**Tiempo(s)**
A12	80	79 %	0.318466	77 %	0.311628
A15	50	87 %	0.027735	91 %	0.031012
A16	60	82 %	0.287754	79 %	0.280707

En el algoritmo 9 se muestra la estructura general de los algoritmos diseñados de búsqueda tabú. Como complemento al algoritmo 9, y referente a la línea 11 de su pseudocódigo, se considera el algoritmo 10, en el cual se detalla la forma en que se construyen y controlan los vecindarios de una solución particular del problema de optimización. El algoritmo 10 recibe como parámetro la solución vigente en el algoritmo de BT, que se denota SolActual, y es a ella a quién se le construye el vecindario respectivo. Además, en la línea 5 del algoritmo 10 se indica la aplicación del algoritmo de k-medias a cada vecino generado en la construcción del vecindario.

En las secciones 2.3.3 y 2.3.4 se detallarán los aspectos propios de los algoritmos de búsqueda tabú por transferencias y mediante movimiento de centros de gravedad.

Algoritmo 9 Algoritmo de BT implementado

Entrada: M: número máximo de iteraciones, CantidadVecinos y ReiniciarSolActualCada.

1: SolActual ← solución inicial generada en forma aleatoria.
2: Iteraciones ← 0.
3: ContadorSinMejoras ← 0.
4: **mientras** Iteraciones $< M$ **hacer**
5: Iteraciones ← Iteraciones+1.
6: ContadorSinMejoras ← ContadorSinMejoras+1.
7: **si** ContadorSinMejoras mód ReiniciarSolActualCada $= 0$ **entonces**
8: Reinicie SolActual en forma aleatoria.
9: Reinicie la lista tabú.
10: **fin si**
11: Genere un vecindario V de SolActual, de tamaño CantidadVecinos.
12: Determine el mejor vecino aceptable en V. Debe respetarse si un vecino es tabú o no y considerar el criterio de aspiración.
13: **si** MejorVecino.Inercia $<$ MejorInerciaDeLaHistoria **entonces**
14: MejorSolucion ← MejorVecino.
15: ContadorSinMejoras ← 0.
16: **fin si**
17: SolActual ← MejorVecino.
18: **fin mientras**
19: **retornar** MejorSolucion.

Algoritmo 10 Algoritmo para construir el vecindario en BT

Entrada: CantidadVecinos y SolActual.

```
1: MenorInercia ← ∞.
2: retornar Mejor vecino admisible en el vecindario.
3: para i ← 1 hasta CantidadVecinos hacer
4:    IndividuoAux ← Generar vecino de SolActual.
5:    Aplique k-medias a IndividuoAux hasta que el algoritmo converja.
6:    si IndividuoAux.Inercia < MenorInercia entonces
7:       si IndividuoAux no es Tabú entonces
8:          MenorInercia ← IndividuoAux.Inercia
9:          MejorVecino ← IndividuoAux
10:         Agregar movimiento a la lista tabú.
11:      si no
12:         si IndividuoAux.Inercia < MejorInerciaDeLaHistoria entonces
13:            MenorInercia ← IndividuoAux.Inercia
14:            MejorVecino ← IndividuoAux
15:            Agregar movimiento a la lista tabú.
16:         fin si
17:      fin si
18:   fin si
19: fin para
```

2.3.3. Búsqueda tabú por transferencias

De manera análoga a como se consideró en los algoritmos de sobrecalentamiento simulado, las soluciones factibles se codificaron mediante el vector VClasificacion definido como:

$$\mathsf{VClasificacion} = (c_1, \ldots, c_n)\,,$$

tal que $c_i \in \{1, \ldots, K\}$ y denota la clase a la que se asignará el individuo x_i de X.

La forma en que se genera un vecino es por transferencia de un individuo de una clase a otra. Para efectos de actualizar la inercia intraclases y la matriz de centros de gravedad del vecino generado, a partir de los atributos respectivos de la solución actual, también se emplearon las fórmulas de actualización expuestas en el teorema 4.

En este algoritmo la lista tabú corresponde a una estructura matricial de memoria, de longitud $n \times L_T$, que tiene la forma:

$$\mathsf{ListaTabú} = \begin{pmatrix} t_{11} & t_{12} & \ldots & t_{1,L_T} \\ t_{21} & t_{22} & \ldots & t_{2,L_T} \\ \vdots & \vdots & \vdots & \vdots \\ t_{n1} & t_{n2} & \ldots & t_{n,L_T} \end{pmatrix},$$

donde la entrada t_{ij} de dicha matriz, $i \in \{1, \ldots n\}$ y $j \in \{1, \ldots L_T\}$, satisface que $t_{ij} \in \{1, \ldots K\}$, e indica que el individuo x_i de X perteneció recientemente a la clase[2] t_{ij}. Además, n es la cantidad de individuos en X que serán clasificados y L_T es un parámetro sobre el cual se detallará posteriormente.

Por lo tanto, la matriz anterior puede visualizarse como una matriz de listas tabú, de

[2]En esta forma de codificar, lo importante es el índice de clase (esto es, el valor que toma t_{ij}); sin embargo, por ejemplo, en una tabla de cinco individuos y tres clases, las clasificaciones $(1, 1, 2, 2, 3)$, $(2, 2, 1, 1, 3)$ y $(3, 3, 2, 2, 1)$ son equivalentes.

tal manera que la fila i de dicha matriz es la memoria que controla las últimas L_T clases a las que ha pertenecido el individuo x_i de X. Dicha memoria es administrada tal y como se muestra en el diagrama de la figura 2.2, el cual hace referencia a la $i-$ésima fila de la matriz de listas tabú.

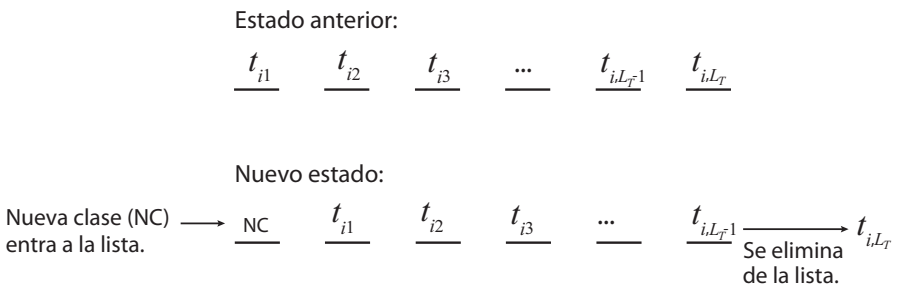


Figura 2.2: Lista tabú en transferencias.

De la figura 2.2 se ilustra que cada vez que el individuo x_i es transferido a una nueva clase NC, entonces se realiza un desplazamiento hacia la derecha de las entradas en la fila i. Esto genera que la clase almacenada en la posición (i, L_T) sea eliminada de la lista, es decir, deja de tener la condición tabú para el individuo x_i. Además, la nueva clase NC es almacenada en la posición $(i, 1)$ de la matriz.

El parámetro L_T representa la longitud de la lista tabú, esto es, la cantidad de clases que para un individuo específico serán designadas como tabú. Es claro que así definido este parámetro, debe darse que $L_T \in \{1, \ldots, K-1\}$. Dado que al inicio de la ejecución del algoritmo un individuo solo ha pertenecido a una clase (en virtud de la configuración con la que inicia el algoritmo), entonces L_T tiene el valor de uno e incrementa una unidad por cada transferencia que se realice, alcanzando un valor máximo de $K-1$. Esto implica que el valor de L_T se asigna y se actualiza de manera automática durante la ejecución, como resultado de las transferencias que se van dando en el proceso. Por este motivo, en el presente estudio no se consideró como un parámetro a optimizar.

Además, es claro que a un individuo específico se le pueden vetar a lo sumo $K-1$ clases. En este sentido, en un caso hipotético podría pasar que un individuo tenga en lista tabú las restantes $K-1$ clases, y ello implicaría que en esa iteración el individuo no puede ser transferido, salvo que el criterio de aspiración lo permita. Sin embargo, esta situación es solventada con el hecho que, como ya se indicó en las generalidades de los algoritmos de búsqueda tabú, la configuración respectiva es reiniciada periódicamente (se reinicia la solución actual desde puntos elegidos aleatoriamente), junto con la lista tabú (para todos los individuos se habilitan nuevamente todas las posibles transferencias).

2.3.4. Búsqueda tabú mediante movimiento de centros de gravedad

Similarmente de como se trató en la heurística de sobrecalentamiento simulado mediante movimiento de centros de gravedad, cada solución factible tiene asociada una estructura del tipo $\{g_1, \dots g_K\}$, que corresponde a la matriz de centroides, donde g_l representa el centroide de la clase C_l, para $l \in \{1, \dots K\}$. Además, para construir el vecindario de una solución factible se siguió la estrategia de mover el centroide de una de las clases. Esto es, suponiendo que se selecciona el centroide l, con $l \in \{1, \dots K\}$, y la posición r de dicho centroide, con $r \in \{1, \dots p\}$, entonces la posición (l, r) de dicho vector se actualiza como:

$$g^*_{lr} = g_{lr} + \Delta_r \quad \text{ó} \quad g^*_{lr} = g_{lr} - \Delta_r,$$

donde Δ_r denota un cierto tamaño de paso usado para ejecutar el movimiento. Así, se suma o resta (bajo una probabilidad) dicho valor, respetando las condiciones $g_{lr} \pm \Delta_r \geq \text{mín}\{x_{1r}, \dots x_{nr}\}$ y $g_{lr} \pm \Delta_r \leq \text{máx}\{x_{1r}, \dots x_{nr}\}$.

A partir de lo anterior, en el presente algoritmo la lista tabú es una estructura com-

puesta por tres vectores de dimensiones $1 \times L_T$:

$$\begin{aligned} \mathsf{CGCambiado} &= (\alpha_1, \ldots, \alpha_{L_T}) \\ \mathsf{Posicion} &= (\beta_1, \ldots, \beta_{L_T}) \\ \mathsf{Signo} &= (\gamma_1, \ldots, \gamma_{L_T}), \end{aligned}$$

donde $\alpha_i \in \{1, \ldots, K\}$, $\beta_i \in \{1, \ldots, p\}$ y $\gamma_i \in \{-1, 1\}$, tal que $i \in \{1, \ldots, L_T\}$. El vector CGCambiado almacena, en la posición correspondiente, el número que codifica el centroide que ha sido cambiado (se guardará el número λ si se modificó el centroide de la clase λ). El vector Posicion almacena, de manera respectiva, la posición que fue cambiada a dicho centroide. Finalmente, el vector Signo se codificó entendiendo que el 1 indica la operación "suma" y el -1 indica la "resta".

En caso que un vecino haya sido generado sumando a la componente respectiva del centroide, entonces en el vector Signo se almacena un -1, indicando que el movimiento prohibido o tabú corresponde a restar en dicha componente, pues dicha operación invierte la transformación realizada. De manera análoga se interpreta en el caso que el vecino haya sido generado mediante una resta, almacenando en este caso un 1.

Los tres vectores anteriores deben ser interpretados en conjunto, posición por posición. Por ejemplo, si $\alpha_i = 2$, $\beta_i = 4$ y $\gamma_i = 1$, para algún $i \in \{1, \ldots, L_T\}$, entonces esto indica que uno de los movimientos que es considerado tabú en la iteración respectiva, corresponde a sumar en la componente número cuatro del centroide de la clase dos.

De manera análoga a como se implementó la lista tabú en el algoritmo de búsqueda tabú por transferencias, en presencia de la construcción de un vecino mediante movimiento de centros de gravedad, las entradas de los vectores CGCambiado, Posicion y Signo son desplazadas hacia la derecha. Provocando que el movimiento almacenado en la posición

$(1, L_T)$ de los tres vectores, sea eliminado como tabú. Adicionalmente, el nuevo movimiento es registrado, en sus tres atributos, en la posición respectiva $(1, 1)$ de los tres vectores.

En un principio la lista tabú está vacía, esto es, los vectores CGCambiado, Posicion y Signo no registran información. Cuando se tiene el mejor vecino factible tomado del vecindario de la solución inicial, entonces $L_T = 1$, esto es, se redimensionan los vectores a tamaño 1×1 y se registra la información respectiva en cada uno de ellos. En la siguiente iteración $L_T = 2$, por ende la dimensión de los vectores será 1×2 para registrar en la segunda posición el movimiento anterior, mientras que en la primera posición se almacena el movimiento actual.

Siguiendo con esta lógica, se nota que L_T incrementa una unidad por cada movimiento registrado. Este incremento podría alcanzar hasta un valor máximo de $2pK$, que corresponde al mayor valor posible de movimientos que, bajo la estrategia expuesta, pueden ser considerados tabú. En la práctica, es poco probable que L_T llegue a alcanzar ese valor máximo, dado que periódicamente se reinicia la solución factible presente en una determinada iteración, junto con la respectiva lista tabú (ver las líneas 8 y 9 del algoritmo 9). En virtud de lo anterior, L_T no fue considerado como un parámetro a optimizar.

2.4. Exploración de la variabilidad: optimización de parámetros

En cada una de las heurísticas implementadas, existen conjuntos de parámetros que deben ser ajustados para su funcionamiento óptimo. La buena escogencia de estos parámetros puede hacer que la eficiencia del método aumente considerablemente con respecto a otro juego de parámetros. En cada heurística existen rangos recomendados para cada parámetro, sin embargo, no existe garantía de que los valores en los rangos sean lo más adecuados y de serlos, es importante estudiar cuales de los valores en los rangos garantizan los mejores resultados.

Por otro lado, las mejoras o pequeñas variantes que se hagan de los algoritmos, pueden hacer variar la calidad de los parámetros escogidos, es decir, un juego de parámetros que se consideran buenos, puede no serlo cuando en la heurística se realizan algunas variaciones tal como en este caso la aplicación de $k-$medias. Por esta razón es indispensable realizar un estudio a profundidad para lograr determinar un juego de parámetros que generen resultados aceptables.

Antes de continuar con el desarrollo del proceso de optimización de parámetros, es necesario entender lo que se conocerá como una malla en una región rectangular definida en $\mathbb{R}$, $\mathbb{R}^2$, $\mathbb{R}^3$ y $\mathbb{R}^4$.

Definición 4 **(Malla de un intervalo real)** *Sea $I = [a, b]$ con $a < b$ un intervalo real y sea n un número natural, se define una malla de I en $n + 1$ puntos, como el conjunto $\mathcal{M}$ definido por: $\mathcal{M} = \{x_0, x_1, x_2, x_3, \ldots, x_n\}$ donde $x_0 = a$, $x_n = b$ y $x_{i-1} < x_i$ para $i = 1, 2, 3, \ldots n$.*

Además, una malla $\mathcal{M} = \{x_0, x_1, x_2, x_3, \ldots, x_n\}$ se dice regular, si se cumple que $x_i -$

$x_{i-1} = x_j - x_{j-1}$ *para todo* $i = 1, 2, 3, \ldots, n$ *y* $j = 1, 2, 3, \ldots, n$.

Finalmente, se define el paso de una malla regular como la distancia que existe entre dos datos consecutivos de la malla.

Notación: Para denotar una malla $\mathcal{M}$ de un intervalo $[a, b]$ en $n+1$ puntos se empleará la notación

$$\mathcal{M} = \{a = x_0 < x_1 < x_2 < x_3 < \cdots < x_n = b\}.$$

Esto evita tener que explicitar las demás condiciones sobre los elementos de $\mathcal{M}$.

DEFINICIÓN **5 (Malla de una región rectangular en $\mathbb{R}^2, \mathbb{R}^3$ y $\mathbb{R}^4$)** *Sea* $D = I_1 \times I_2$ *una región rectangular en* $\mathbb{R}^2$ *con* $I_1 = [a, b]$ *y* $I_2 = [c, d]$. *Sean* $\mathcal{M}_1$ *y* $\mathcal{M}_2$ *malla de los intervalos* I_1 *y* I_2 *respectivamente tal que:*

$$\begin{aligned} \mathcal{M}_1 &= \{a = x_0 < x_1 < x_2 < \cdots < x_n = b\} \\ \mathcal{M}_2 &= \{c = y_0 < y_1 < y_2 < \cdots < y_m = d\}. \end{aligned}$$

Se define la malla $\mathcal{M}$ *de la región* D *por:* $\mathcal{M} = \mathcal{M}_1 \times \mathcal{M}_2$. *Donde,* $(x_i, y_j) \in \mathcal{M}$, *si y solo si,* $x_i \in \mathcal{M}_1$ *y* $y_j \in \mathcal{M}_2$.

De manera análoga se define una malla $\mathcal{M}$ *de las regiones:*

- $I_1 \times I_2 \times I_3 \subseteq \mathbb{R}^3$ *como el conjunto formado por* $\mathcal{M} = \mathcal{M}_1 \times \mathcal{M}_2 \times \mathcal{M}_3$ *donde* $\mathcal{M}_1, \mathcal{M}_2$ *y* $\mathcal{M}_3$ *son mallas de los intervalos* I_1, I_2 *y* I_3 *respectivamente;*

- $I_1 \times I_2 \times I_3 \times I_4 \subseteq \mathbb{R}^4$ *como el conjunto formado por* $\mathcal{M} = \mathcal{M}_1 \times \mathcal{M}_2 \times \mathcal{M}_3 \times \mathcal{M}_4$ *donde* $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$ *y* $\mathcal{M}_4$ *son mallas de los intervalos* I_1, I_2, I_3 *y* I_4 *respectivamente.*

Finalmente, se dice que una malla es regular si el paso en cualquiera de sus componentes es el mismo. En este caso, al tamaño de paso se le denominará grosor de la malla.

En el presente trabajo se realiza una optimización de los parámetros de las heurísticas denominadas: algoritmos genéticos, enjambres de partículas y colonias de hormigas; mismas aplicadas al problema de particionamiento con datos cuantitativos. El procedimiento con el cual se realiza esta labor, consiste en la construcción de una malla del espacio de búsqueda de los parámetros, para luego recorrer los puntos de la malla y determinar en éstos el porcentaje de atracción a la mejor solución conocida.

El proceso de optimización de parámetros se realizó por medio de las tablas de datos Tabla-105-K(7)Card($\neq$)DS($\neq$) y Tabla-525-K(7)Card($\neq$)DS($\neq$), de ahora en adelante se hará referencia a ellas como las tablas base. Sin embargo, en algunas heurísticas fue necesario realizar una optimización en otras tablas para poder ajustar mejor los parámetros dada la poca evidencia de calidad que mostraron los parámetros determinados únicamente con las dos tablas base.

2.4.1. Algoritmos genéticos

Los parámetros a ajustar en la heurística de algoritimos genéticos corresponde a los porcentajes de mutación p_m y el porcentaje de cruzamiento p_c. Estos parámetros al ser probabilidades deben cumplir que $0 \leq p_c, p_m \leq 1$, sin embargo, los valores $p_c = 0$ y $p_m = 0$ eliminan por completo el operador de cruce y de mutación del algoritmo por lo que los mismos no serán considerados durante la optimización. Razón por la cual se considerará que tanto p_c como p_m están ambos en el intervalo $]0, 1]$ y la región de búsqueda queda definida por $\mathcal{R} =]0, 1] \times]0, 1]$ donde la primera coordenada corresponde a p_c y la segunda a p_m.

Para el estudio de los parámetros subóptimos en la heurística algoritmos genéticos, el

número de instancias se fijo en 30, la aplicación de $k-$medias se hizo cada 4 iteraciones y el algoritmo esperó un máximo 30 iteraciones sin mejora como método de paro.

Para cada una de las tablas estudiadas se realizó una optimización preliminar, con un mallado regular $\mathcal{R}$ haciendo variar cada parámetro de 0.1 a 1 a pasos de 0.1 y calculando el porcentaje de atracción por medio de 100 corridas sucesivas en cada punto de la malla. Posteriormente, se realiza una segunda optimización con una malla regular de grosor 0.05 construida en una subregión producto de lo observado en el proceso preliminar. Para esta segunda malla se realizaron 1000 corridas sucesivas para el cálculo del porcentaje de atracción, ésto fue posible gracias a los bajos tiempos de ejecución reportados para dichas tablas de datos.

Análisis realizado a Tabla-105-K(7)Card($\neq$)DS($\neq$)

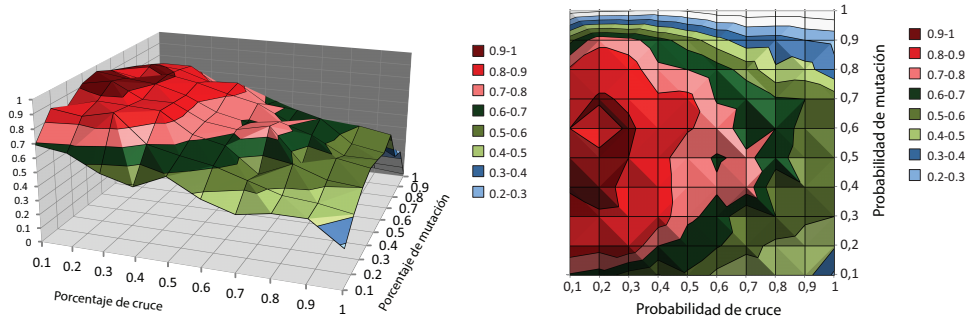
En la figura 2.3(a) se apreciar la superficie generada por el porcentaje de atracción de la primera optimización, en donde claramente se observa una región con valores de atracción superiores al 90 %. En la figura 2.3(b) se observa en detalle los puntos de la malla dentro de esta región, en esta figura se puede visualizar que para que el porcentaje de atracción sea superior al 80 % debe cumplirse que:

- la probabilidad de cruce p_c debe estar entre 0.1 y 0.4;
- la probabilidad de mutación p_m debe estar entre 0.2 y 0.8.

De esta manera, se procede a realizar un afinamiento de la malla en la región $[0.05, 0.55]\times [0.15, 0.85]$. En la figura 2.3(c) se puede observar la superficie generada por el porcentaje de atracción en los puntos de la malla, en la misma se puede visualizar una clara región en donde los porcentajes de atracción son superiores al 90 %. En la figura 2.3(d), se puede

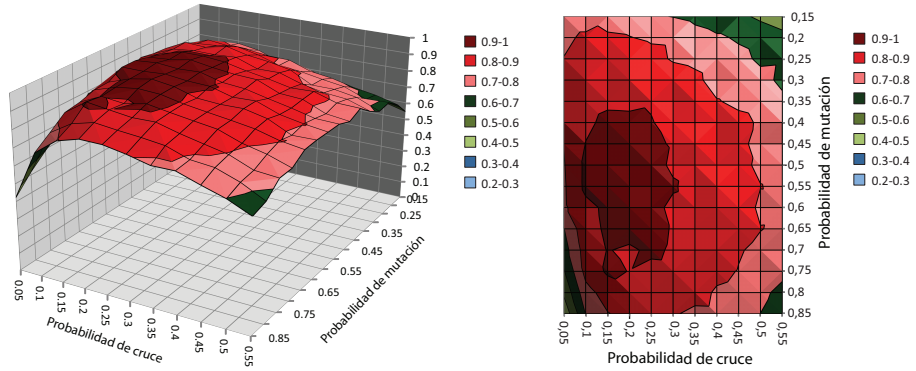
apreciar, con detalle, los puntos de la malla que están dentro de ésta última región, de esta misma figura se puede deducir que:

- el porcentaje de cruce p_c debe variar entre 0.1 y 0.3;
- el porcentaje de mutación p_m debe variar entre 0.4 y 0.7.



(a) Superficie generada por la malla preliminar de grosor 0.1.

(b) Curvas de nivel generada por la malla prelimiar de grosos 0.1.



(c) Superficie generada por la malla secundaria de grosor 0.05.

(d) Curvas de nivel generada por la malla secundaria de grosor 0.05.

Figura 2.3: Superficie generada por el porcentaje de atracción en la tabla de datos Tabla-105-K(7)Card(≠)DS(≠).

Análisis realizado a Tabla-525-K(7)Card(≠)DS(≠)

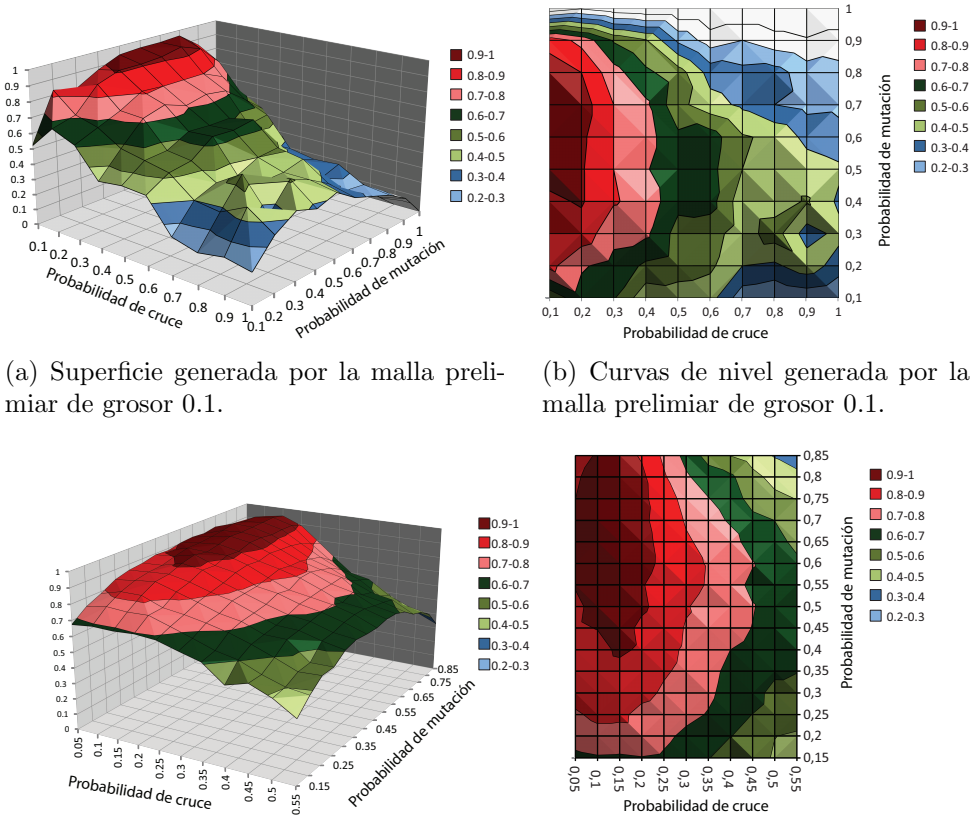
El figura 2.4(a) se observa la superficie generada por el porcentaje de atracción de la primera optimización. Similar a lo ocurrido para los datos anteriores, es posible observar una región cuyos valores de atracción son superiores al 90 %. En la figura 2.4(b) se puede delimitar mejor los punto de la malla contenido en dicha región. En esta misma figura, se deduce que para generar porcentajes de atracción superiores al 80 % debe cumplirse que:

- la probabilidad de cruce p_c debe estar entre 0.1 y 0.3;
- la probabilidad de mutación p_m debe estar entre 0.4 y 0.8

De manera análoga a lo realizado en la optimización de la tabla de datos anterior, se procedió a realizar un afinamiento de la malla en la región $[0.05, 0.55] \times [0.15, 0.85]$. Producto de este afinamiento se tiene la figura 2.4(c), en la cual se observa claramente una región cuyos porcentajes de atracción son superiores al 90 %, los puntos de la malla dentro de esta región se puede visualizar mejor en la figura 2.4(d), de la misma se deduce que:

- el porcentaje de cruce p_c debe variar entre 0.05 y 0.2;
- el porcentaje de mutación p_m debe variar entre 0.45 y 0.85.

Finalmente, con fundamento en los dos análisis hechos para las tablas base, se decidió tomar como probabilidad de cruce $p_c = 0.15$, mientras que en la probabilidad de mutación se tomó como $p_m = 0.65$, dicha escogencia obedecen a puntos de la malla en el interior de las regiones que reporta buenos porcentajes de atracción en ambas tablas.



(a) Superficie generada por la malla prelimiar de grosor 0.1.

(b) Curvas de nivel generada por la malla prelimiar de grosor 0.1.

(c) Superficie generada por la malla secundaria de grosor 0.05.

(d) Curvas de nivel generada por la malla secundaria de grosor 0.05.

Figura 2.4: Superficie generada por el porcentaje de atracción en la tabla de datos Tabla-525-K(7)Card(≠)DS(≠).

2.4.2. Enjambres de partículas

El algoritmo de enjambres de partículas posee básicamente tres parámetros a ajustar: el parámetro de tendencia α que corresponde al peso que posee la velocidad anterior en el cálculo de la nueva velocidad, c_1 que representa la influencia máxima que posee la mejor experiencia vivida por la partícula (parámetro cognitivo) y c_2 que representa el máximo peso que posee la posición del líder del enjambre en el cálculo de la nueva velocidad (parámetro social).

Otro enfoque es considerar un cuarto parámetro w definido por $w = c_1 + c_2$, viendo en w la suma de los pesos relativos del parámetro cognitivo y social, tal como se pudo observar en análisis de divergencia mencionado en la página 44 de este documento.

En base a lo anterior, se decidió optimizar los parámetros α, c_1 y w. Cada uno de estos parámetros fueron estudiados en los siguientes rangos:

- α: se consideró en el conjunto $[-2, 2]$;
- c_1: se consideró en el conjunto $[0, 4.5]$;
- w: se consideró en el conjunto $[0.5, 5]$.

A continuación se presentan, para cada una de las tablas base, los datos más relevantes producto del proceso de optimización de parámetros. En las ejecuciones realizadas en este proceso, el número de partículas se tasó en 10, la aplicación del $k-$medias se realizó cada 3 iteraciones sin mejora y el algoritmos esperó 10 iteraciones sin mejora en el método de paro.

Análisis realizado a Tabla-105-K(7)Card($\neq$)DS($\neq$)

En esta primera tabla de datos se realizó un proceso preliminar de optimización por medio de una malla regular con un grosor de 0.5. Para el cálculo del porcentaje de atracción se completaron un total de 100 ejecuciones sucesivas, ésto con el objetivo de disminuir, en alguna medida, el tiempo de ejecución del proceso de optimización, puesto que el mismo se torna extenso debido a la gran cantidad de puntos que posee la malla producto de la cantidad de dimensiones.

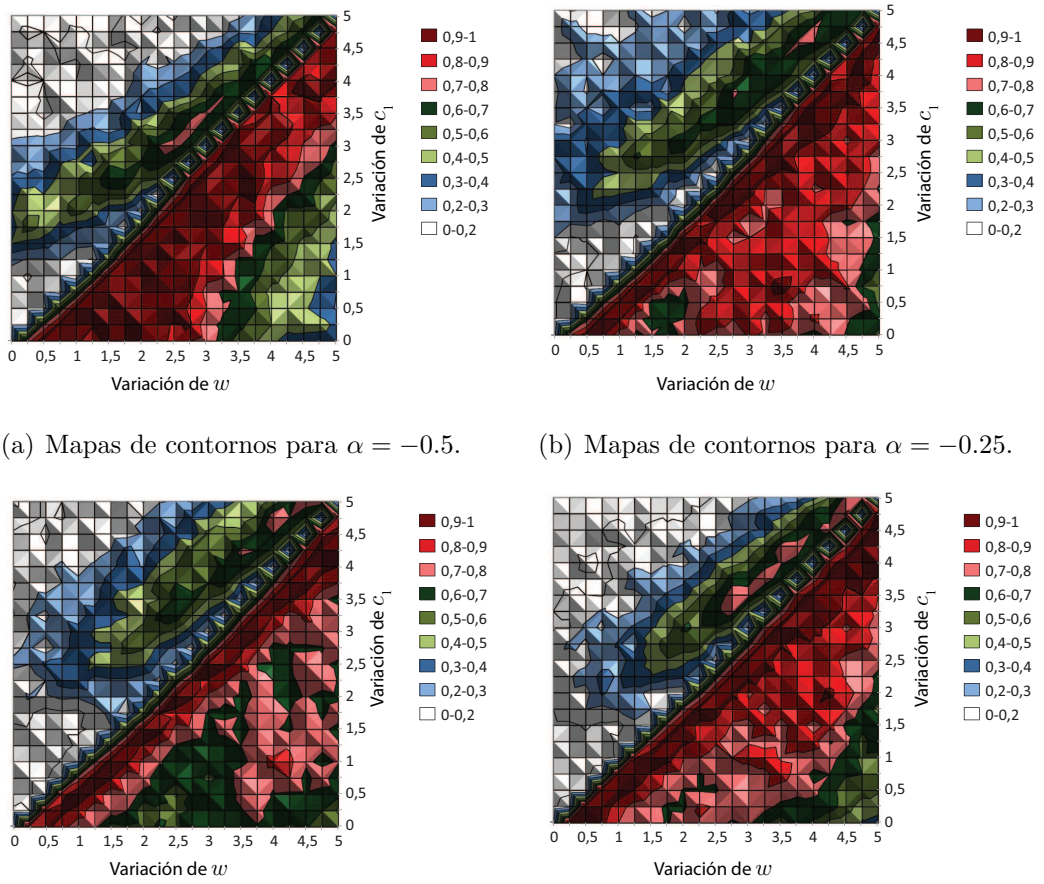
Los resultados obtenidos de este primer proceso de optimización de parámetros se presentan por medio de mallas bidimensionales producto de fijar el valor de α y variando, en sus respectivos rangos, los valores para c_1 y w.

Producto de este primer análisis, se realiza un segundo proceso de optimización por medio de un afinamiento de la malla utilizando un grosor de 0.25 y 500 ejecuciones sucesivas para el cálculo del porcentaje de atracción. Para este nuevo proceso, los valores de α fueron escogidos como aquellos que resultaron ser más prometedores en el análisis preliminar.

De la observación del conjunto de las imágenes presentes en la figura 2.5, producto del primer análisis, se observan un comportamiento marcado y consistente en todas las gráficas presentadas que no depende del valor de α. En los mapas de contorno se observa que la región en la cual se reportan los mejores porcentajes de atracción, para esta primera tabla de datos, se extiende paralela a una de las diagonales de la malla. La característica más notoria de los puntos de la malla dentro de la región en donde se reportaron los mejores porcentajes de atracción corresponde a una relación entre el parámetro w y el parámetro c_1, dada por

$$w = c_1 + 0.5. \tag{2.4}$$

Las imágenes presentadas en la figura 2.6, son producto del segúndo análisis, en ellas se muestran las mallas bidimensionales producidas al fijar el valor $\alpha = -0.5$, $\alpha = -0.25$, $\alpha = 0.25$ y $\alpha = 0.5$. Es posible observar que la relación 2.4 permanece invariante para esta primera tabla de datos.



(a) Mapas de contornos para $\alpha = -0.5$.

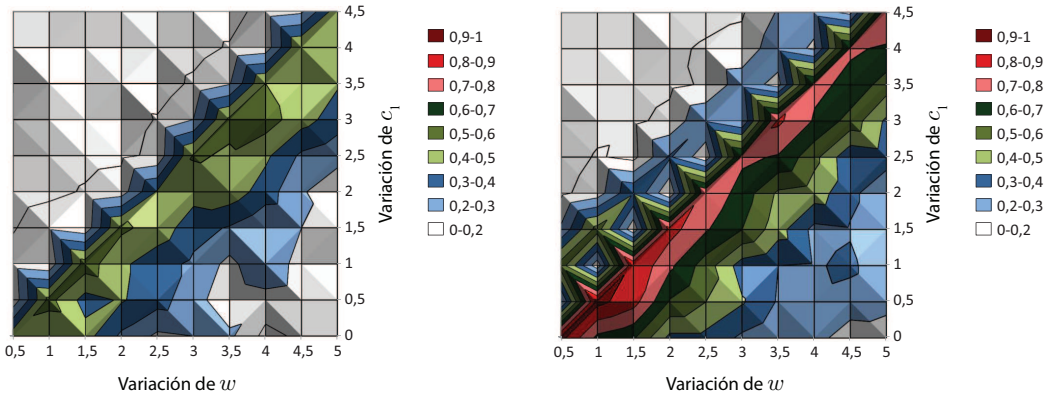
(b) Mapas de contornos para $\alpha = -0.25$.

(c) Mapas de contornos para $\alpha = 0.25$.

(d) Mapas de contornos para $\alpha = 0.5$.

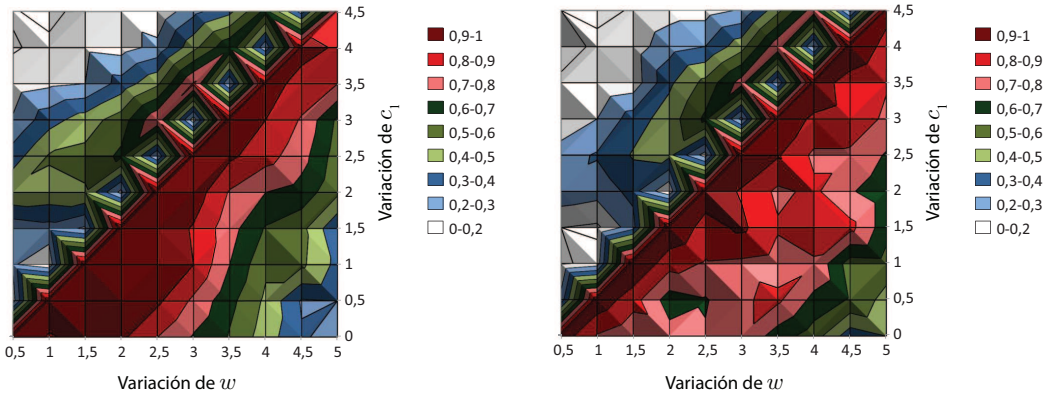
Figura 2.6: Mapa de contornos generado por el porcentaje de atracción para un α fijo y una malla regular bidimensional de 0.25 de grosor, producto de la variación de los parámetros w y c_1.

Con respecto al valor del parámetro α es posible observar un comportamiento marcado, principalmente, en los mapas de contorno del primer análisis. Se puede notar que para



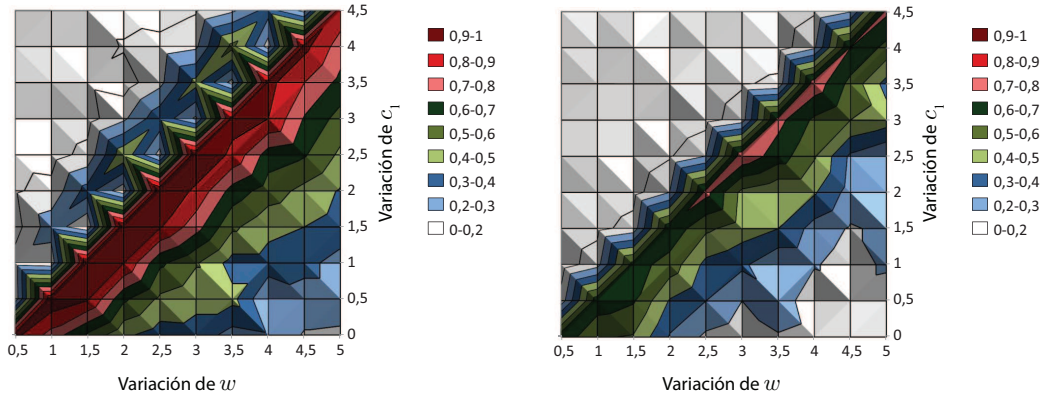
(a) Mapas de contornos para $\alpha = -2$.

(b) Mapas de contornos para $\alpha = -1$.



(c) Mapas de contornos para $\alpha = -0.5$.

(d) Mapas de contornos para $\alpha = 0.5$.



(e) Mapas de contornos para $\alpha = 1$.

(f) Mapas de contornos para $\alpha = 2$.

Figura 2.5: Mapa de contornos generado por el porcentaje de atracción para un α fijo y una malla regular bidimensional producto de la variación de los parámetros w y c_1.

valores de α cercanos a cero, ver figuras 2.5(c) y 2.5(d), los porcentajes de atracción para los puntos de las mallas que satisfacen la relación 2.4 se incrementan considerablemente. Por su parte, lo valores de α que tienden a estar alejados de cero, producen que el porcentaje de atracción en los punto de las mallas que satisfacen 2.4 tiendan a la baja.

Análisis realizado a Tabla-525-K(7)Card($\neq$)DS($\neq$)

Para esta segunda tabla de datos se realiza un proceso análogo al análisis preliminar realizado para la tabla anterior. Fijando diferentes valores del parámetro α y construyendo mallas bidimensionales de groso 0.5 mediante la variación de c_1 y w. Los porcentajes de atracción para este análisis fue calculado por medio de 100 ejecuciones sucesivas.

En la figura 2.7 es posible observar que la relación 2.4 reaparece para esta nueva tabla de datos. También, el comportamiento del parámetro α es similar al presentado para la tabla de datos Tabla-105-K(7)Card($\neq$)DS($\neq$), pudiendo observarse, para valores de α alejados de cero, que el porcentaje de atracción en los puntos de la malla que cumplen la relación 2.4 tiende a disminuir, no siendo así para valores de α cercanos a cero.

Por esta razón, según lo que se logra observar, es razonable seleccionar como valor para el parámetro c_1 el valor $w - 0.5$ para cualquier $w \in [0.5, 5]$. Debido a la cantidad de posible valores que se pueden seleccionar para los parámetros w y c_1, se procedió a realizar la escogencia de dos juegos de valores, para dichos parámetros, mismos que se presenta a continuación.

- $w = 1.75$ y $c_1 = 1.25$.
- $w = 4.25$ y $c_1 = 3.75$.

Por su parte, el parámetro α parece no depender de la escogencia de w y c_1, en el

sentido siguiente, diferentes valores de α producen diferentes porcentajes de atracción disminuyendo éstos a medida que α se aleje de cero, sin embargo, la región en donde se localiza los mejores valores del porcentaje de atracción se localiza siempre en los puntos de la malla que satisface la relación 2.4, ver figuras 2.5 y 2.7.

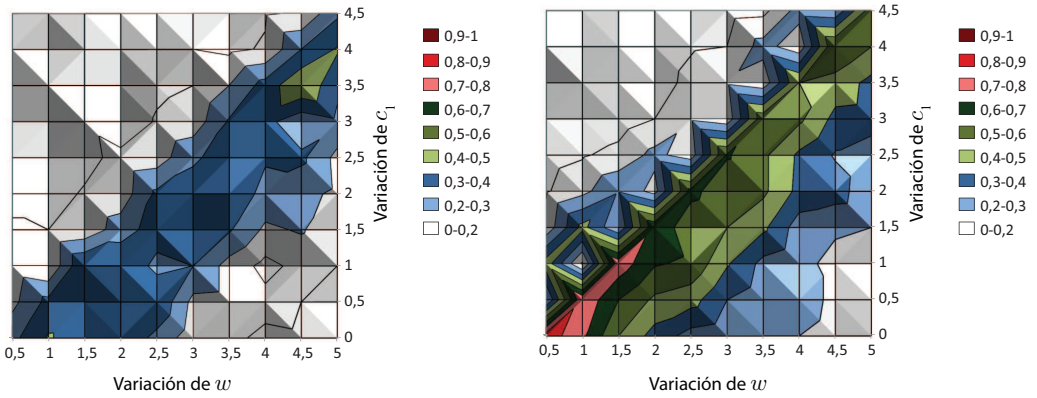
Del análisis realizado se desprende que valores del parámetro α cercanos a cero producen mejores resultados que aquellos valores que se encuentran un poco más alejados. Preliminarmente se puede recomendar tomar $\alpha \in [-1, 1]$. Sin embargo es necesario realizar un estudio más a fondo con el objetivo de poder precisar mejor el valor de este parámetro.

Optimización del parámetro α o parámetro de tendencia

Una vez realizado la escogencia de los parámetros w y c_1 se puede hacer un estudio específico sobre el comportamiento del porcentaje de atracción para diferentes valores de α. Para una mejor confianza de la escogencia de dicho parámetro se realizaron corridas para las tablas que presentan mayor dificultad o resistencia a alcanzar el $100\,\%$ de atracción.

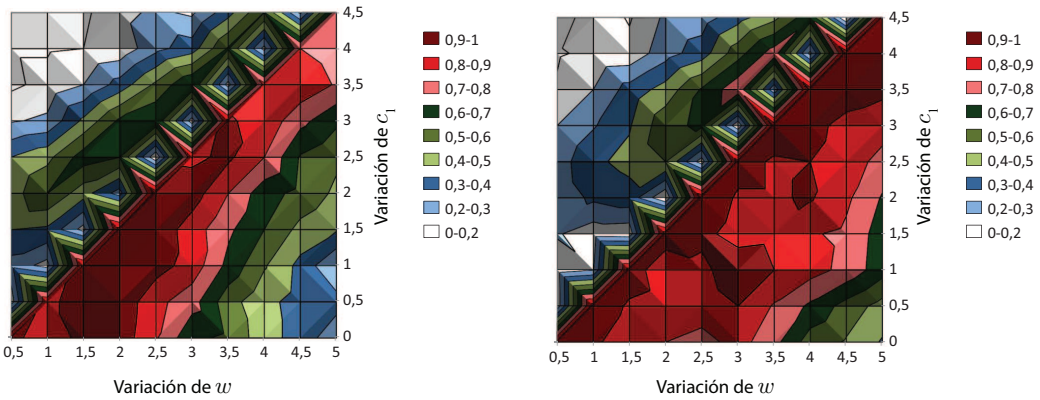
- TA15: Tabla-105-K(7)Card($\neq$)DS($\neq$)
- TA16: Tabla-525-K(7)Card($\neq$)DS($\neq$)
- TB3: Tabla-1050-K(7)Card($=$)DS($\neq$)
- TB4: Tabla-2100-K(7)Card($=$)DS($\neq$)
- TB7: Tabla-1050-K(7)Card($\neq$)DS($\neq$)
- TB8: Tabla-2100-K(7)Card($\neq$)DS($\neq$)

Para el primer juego de parámetros $w = 1.75$ y $c_1 = 1.25$ se hizo variar el parámetro $\alpha \in [-2, 2]$ a pasos de 0.125. Para este proceso se mantuvo el número de partículas en 10,



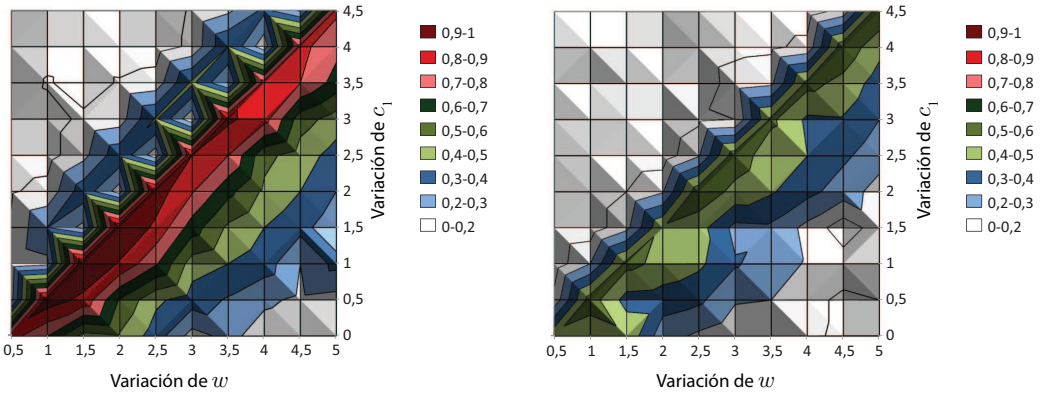
(a) Mapas de contornos para $\alpha = -2$.

(b) Mapas de contornos para $\alpha = -1$.



(c) Mapas de contornos para $\alpha = -0.5$.

(d) Mapas de contornos para $\alpha = 0.5$.



(e) Mapas de contornos para $\alpha = 1$.

(f) Mapas de contornos para $\alpha = 2$.

Figura 2.7: Mapa de contornos generado por el porcentaje de atracción para un α fijo y una malla regular bidimensional producto de la variación de los parámetros w y c_1.

la aplicación de $k-$medias se realizó cada 3 iteraciones sin mejora, el algoritmo espero 10 iteraciones sin mejoras antes de salir y se realizaron 500 corridas sucesivas para el cálculo del porcentaje de atracción.

Las curvas generadas por el porcentaje de atracción para cada una de las tablas se presentan en la figura 2.8. En esta misma figura se puede observar que para la mayoría de las tablas estudiadas un valor de α entre -1 y 1 parece ser bueno, sin embargo para la tabla TB8 el valor escogido deberá estar entre 0.5 y 1. Razón por la cual el valor $\alpha = 0.7$ resultó ser una de las mejores opciones a escoger.

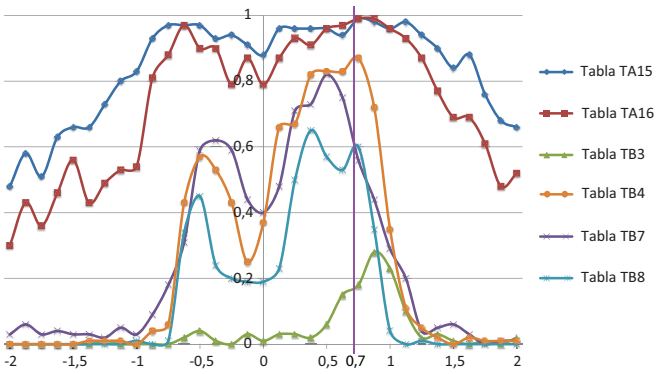


Figura 2.8: Curvas generadas por el porcentaje de atracción producto de la variación del parámetro $\alpha \in [-2, 2]$ en una malla unidimensional de grosor 0.125, con $w = 1.75$ y $c_1 = 1.25$.

Para el caso del segundo juego de parámetros, $w = 4.25$ y $c_1 = 3.75$, la curvas se generaron fijando el número de partículas en 15, la aplicación de $k-$medias se realizó cada 3 iteraciones sin mejora, el algoritmo esperó 15 iteraciones sin mejora en el método de paro y se realizó un total de 500 ejecuciones sucesivas para el cálculo de porcentaje de atracción. Las curvas generadas se presentan en la figura 2.9. En la misma se puede observar que un buen candidato es $\alpha = 0.3$ razón por la cual fue escogido como mejor parámetro.

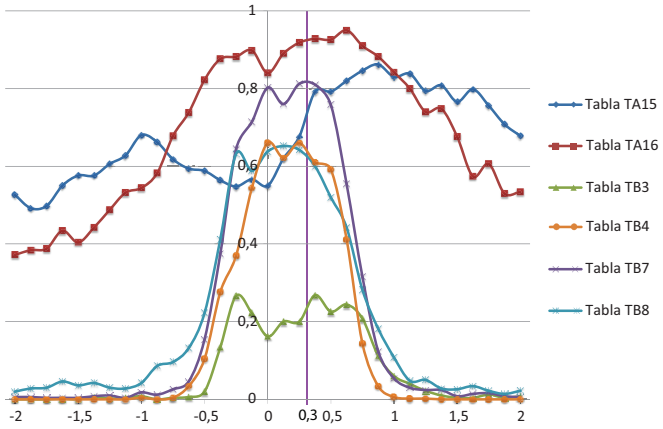


Figura 2.9: Curvas generadas por el porcentaje de atracción producto de la variación del parámetro $\alpha \in [-2, 2]$ en una malla unidimensional de grosor 0.125, con $w = 4.25$ y $c_1 = 3.75$.

De este modo se dá por finalizado el proceso de optimización de parámetros para el algoritmo basado en enjambres de partículas. Del proceso se desprende dos posibles conjuntos de parámetros los cuales generan buenos resultados, estos mismos se presentan en la tabla 2.10.

Tabla 2.10: Resumen de mejores parámetros para enjambre de partículas.

Primer conjunto de parámetros		Segundo conjunto de parámetros	
Parámetro	**Valor propuesto**	**Parámetro**	**Valor propuesto**
w	1.75	w	4.25
c_1	1.25	c_1	3.75
α	0.7	α	0.3

2.4.3. Búsqueda tabú por transferencias

En la figura 2.10 se muestra el panel de control del algoritmo de búsqueda tabú por transferencias.

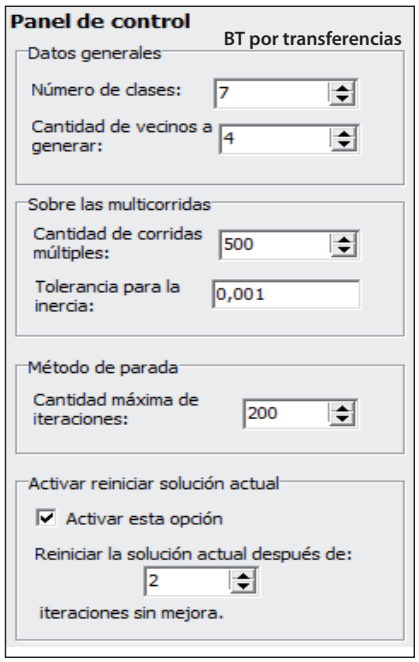


Figura 2.10: Panel de control del algoritmo de BT por transferencias.

Tal y como se explicó en la sección 2.3, en los algoritmos diseñados de búsqueda tabú, la lista tabú es una estructura cuya naturaleza y dimensión se ajusta de manera automática. Por lo tanto, no se considera ningún parámetro que represente la longitud de dicha lista y que tenga que ser optimizado.

Por otra parte, el parámetro CantidadVecinos (véase la opción "Cantidad de vecinos a generar" en la figura 2.10) tampoco debe ser ajustado, debido a la naturaleza esperable de que a mayor cantidad de vecinos, mayor exploración del vecindario de la solución factible vigente en una determinada iteración. Así, se espera que a mayor valor de CantidadVecinos,

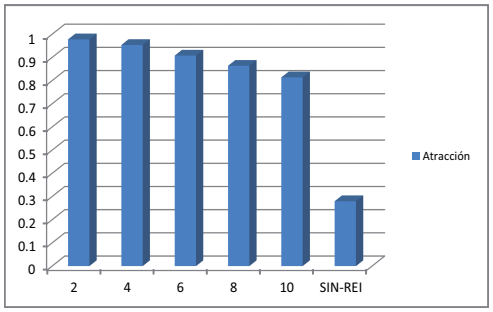


Figura 2.11: Resultados obtenidos al variar el parámetro ReiniSolActualCada en la heurística BT por transferencias, en la tabla de 105 individuos.

mayor probabilidad de que el algoritmo converja al óptimo global.

En general, el único parámetro optimizado en los algoritmos de búsqueda tabú fue ReiniSolActualCada. Se determinó analizar el rendimiento del algoritmo tomando ReiniSolActualCada $\in \{2, 4, 6, 8, 10\}$. Inclusive, se realizó la comparación con el caso en que dicha opción fuera desactivada, esto es, que la solución actual en una iteración específica nunca fuera reiniciada desde puntos aleatorios del espacio de búsqueda (el porcentaje de atracción respectivo se representa con la barra etiquetada SIN-REI, en los gráficos mostrados en las figuras 2.11 y 2.12).

En el gráfico de barras mostrado en la figura 2.11 se muestran los resultados de la aplicación de la presente heurística a la tabla de 105 individuos. En dicho gráfico se evidencia que el algoritmo de búsqueda tabú por transferencias presenta un mejor rendimiento cuando se aplica la estrategia asociada al parámetro ReiniSolActualCada. Además, se determina que los mejores resultados se obtienen al fijar ReiniSolActualCada $= 2$.

Por su parte, en la figura 2.12 se muestran los resultados de la aplicación del algoritmo de búsqueda tabú por transferencias a la tabla de 525 individuos. De la misma forma que en la tabla de 105 individuos, se tienen mejores resultados en la aplicación del algoritmo si se toma el parámetro ReiniSolActualCada con un valor de 2. Adicionalmente, en la barra

etiquetada SIN-REI se nota de una manera más marcada la pérdida de calidad cuando se desactiva la opción de reiniciar la solución actual. En resumen, se tiene la conveniencia de aplicar dicha estrategia y por lo tanto así se hizo en las ejecuciones finales de las demás tablas de control. Los resultados correspondientes serán detallados en el capítulo 3.

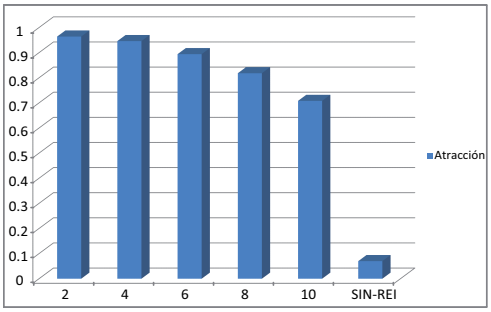


Figura 2.12: Resultados obtenidos al variar el parámetro ReiniSolActualCada en la heurística BT por transferencias, en la tabla de 525 individuos.

2.4.4. Búsqueda tabú mediante movimiento de centros de gravedad

El panel de control de la presente heurística coincide con el mostrado en la figura 2.10. De la misma forma, el único parámetro a ajustar corresponde a ReiniSolActualCada y de acuerdo con las pruebas realizadas, dicho parámetro generó mejores resultados si su valor es tomado del conjunto $\{2, 4, 6, 8, 10\}$ (esto se puede inferir de los gráficos mostrados en las figuras 2.13 y 2.14).

En la figura 2.13 se muestran los resultados obtenidos al aplicar el algoritmo a la tabla de 105 individuos. Similarmente, en la figura 2.14 se puede visualizar el comportamiento de la convergencia del algoritmo hacia el óptimo global, en la tabla de 525 individuos.

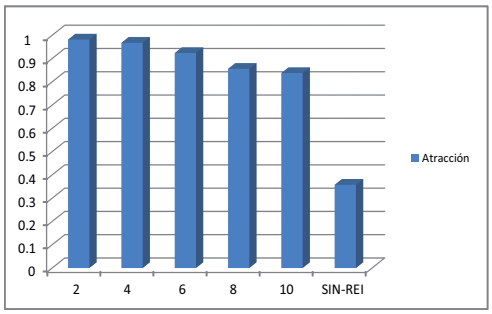


Figura 2.13: Resultados obtenidos al variar el parámetro ReiniSolActualCada en la heurística BT mediante movimiento de CG, en la tabla de 105 individuos.

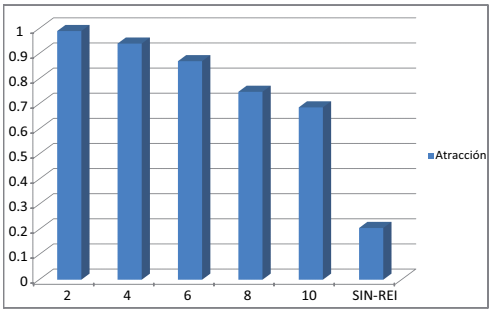


Figura 2.14: Resultados obtenidos al variar el parámetro ReiniSolActualCada en la heurística BT mediante movimiento de CG, en la tabla de 525 individuos.

En ambos casos se nota, al igual que como sucedió en la heurística de búsqueda tabú por transferencias, la conveniencia de reiniciar periódicamente la solución actual desde puntos aleatorios del espacio de búsqueda. Específicamente, existe una coincidencia en tomar ReiniSolActualCada $= 2$ como el valor óptimo para ese parámetro. En efecto, así se hizo y los resultados obtenidos se muestran en el capítulo 3.

Capítulo 3

Apartado 10: Resultados

Análisis del rendimiento y comparación de los algoritmos

En el presente capítulo se mostrarán los resultados obtenidos al aplicar las heurísticas diseñadas a las diferentes tablas de datos. En todos los casos se considera el parámetro TolInercia, que hace referencia a la tolerancia que se tendrá sobre las inercias intraclases obtenidas en las múltiples ejecuciones que se hacen para una misma tabla, y con el objetivo de reportar el porcentaje de atracción. Esto es, se considera que una heurística atinó en una ejecución al óptimo global de la tabla respectiva, si la diferencia absoluta entre la inercia de referencia y la inercia calculada es menor que TolInercia. En todos los casos se consideró TolInercia= 0.001.

Como se indicó en el capítulo anterior, y considerando la interpretación realizada sobre TolInercia, los porcentajes de atracción fueron calculados en función de la cantidad de veces que cada heurística atinó al óptimo global en 500 ejecuciones (en los diferentes formularios, esta opción se denomina "corridas múltiples"). Los tiempos de ejecución serán

reportados en segundos, con precisión a millonésimas de segundo, empleando como separador el símbolo ".". Por ejemplo, el tiempo 1.453214 se debe entender como "un segundo con 453214 millonésimas de segundo".

3.1. Resultados en tablas de referencia

Para facilitar la presentación de los resultados, en la tabla 3.1 se codifican las 16 tablas de datos simulados elaboradas por la M.Sc. Alexia Pacheco y que son citadas en Trejos et al. (2006). Esta codificación será referenciada en las tablas de presentación de resultados.

Tabla 3.1: Codificación de las tablas de referencia.

Código	Características de la tabla	Inercia de referencia
A1	$n = 105$, $K = 3$, Card(=) DS(=)	5.42201131972789
A2	$n = 525$, $K = 3$, Card(=) DS(=)	5.99260745795918
A3	$n = 105$, $K = 7$, Card(=) DS(=)	5.14604458106576
A4	$n = 525$, $K = 7$, Card(=) DS(=)	5.33900848761904
A5	$n = 105$, $K = 3$, Card(=) DS($\neq$)	13.1501508312277
A6	$n = 525$, $K = 3$, Card(=) DS($\neq$)	15.8091025419975
A7	$n = 105$, $K = 7$, Card(=) DS($\neq$)	9.89537854743636
A8	$n = 525$, $K = 7$, Card(=) DS($\neq$)	8.26081280342641
A9	$n = 105$, $K = 3$, Card($\neq$) DS(=)	5.0067792302106
A10	$n = 525$, $K = 3$, Card($\neq$) DS(=)	5.67158162531383
A11	$n = 105$, $K = 7$, Card($\neq$) DS(=)	5.54505587301587
A12	$n = 525$, $K = 7$, Card($\neq$) DS(=)	5.64774256034859
A13	$n = 105$, $K = 3$, Card($\neq$) DS($\neq$)	11.7340341970121
A14	$n = 525$, $K = 3$, Card($\neq$) DS($\neq$)	13.8192625021861
A15	$n = 105$, $K = 7$, Card($\neq$) DS($\neq$)	7.62467182751889
A16	$n = 525$, $K = 7$, Card($\neq$) DS($\neq$)	7.4561026338506

3.1.1. Resultados en algoritmos genéticos y enjambres de partículas

En la presente sección se resumen los resultados obtenidos en la aplicación de las heurísticas de algoritmo genético y enjambre de partículas al problema de particionamiento con datos cuantitativos en las 16 tablas de referencia. En el caso de enjambres de partículas se realizaron dos aplicaciones por cada tabla de datos, uno para cada grupo de parámetros determinados en el capítulo 2.4.

Con el objetivo de que el porcentaje de atracción fuera el más alto posible en un tiempo razonable, en estas dos heurísticas y para cada tabla de datos se ajustó:

- el número de individuos en las poblaciones respectivas (cantidad de agentes): número de instancias en la población a evolucionar en los algoritmo genéticos (Núm. Insta.) y la cantidad de partículas en el enjambre (Núm. parti.);

- el número de iteraciones sin que el método haya reportado una mejora antes de dar finalizado la ejecución del algoritmo, que corresponde a la condición del método de paro;

- la frecuencia con la cual se aplica la mejora de $k-$medias;

En las tablas 3.2 y 3.3 se pueden apreciar los resultados obtenidos en la ejecución para los algoritmos genéticos y enjambres de partículas con los juegos de parámetros determinados en el capítulo 2.4. Para el caso de AG la probabilidad de mutar y de cruce fueron fijados en $p_m = 65\,\%$ y $p_c = 15\,\%$, respectivamente. Para enjambre de partículas se utiliza la codificación EP-1 que indica el uso de la configuración de parámetros $\alpha = 0.7$, $c_1 = 1.25$ y $w = 1.75$, y EP-2 con el juego de parámetros $\alpha = 0.3$, $c_1 = 3.75$ y $w = 4.25$.

Tabla 3.2: Resultados generados por algoritmos genéticos (AG) en las 16 tablas de referencia, en 500 inicializaciones al azar.

Algoritmos Genéticos 100 % de atracción en las 16 tablas				
Tabla código	**Tiempo promedio**	**Núm. Insta.**	**Iter. espera**	**Aplica K-M**
A1	0.001683	5	4	2
A2	0.003879	3	2	2
A3	0.040858	18	18	2
A4	0.139791	16	15	3
A5	0.011994	12	11	2
A6	0.129875	18	18	3
A7	0.556964	73	72	3
A8	0.074325	10	10	3
A9	0.001873	8	4	4
A10	0.005745	4	3	3
A11	0.080337	30	30	3
A12	0.222646	23	15	3
A13	0.003535	8	6	3
A14	0.020404	9	8	3
A15	0.126575	30	30	2
A16	2.093986	90	40	3

Tal y como se indica en las tablas 3.2 y 3.3, para las tablas de referencia, las tres heurísticas (AG, EP-1 y EP-2) lograron un rendimiento del 100 % en todas las tablas.

3.1.2. Resultados en búsqueda tabú

En esta sección se mostrarán los resultados obtenidos en las heurísticas de búsqueda tabú en las tablas de referencia. En la tabla 3.4 se muestran los resultados obtenidos considerando el parámetro ReiniSolActualCada = 2. La columna encabezada con **MaxIte** indica la cantidad máxima de iteraciones que necesitó el algoritmo respectivo para alcanzar el porcentaje de atracción especificado.

Tabla 3.3: Resultados en enjambres de partículas (OEP) para las 16 tablas de referencia, en 500 inicializaciones al azar.

Enjambre de partículas 1 (EP-1) 100 % de atracción en las 16 tablas					**Enjambre de partículas 2 (EP-2)** 100 % de atracción en las 16 tablas				
Tabla código	**Tiempo promedio**	**Núm. parti.**	**Iter. espera**	**Aplica K-M**	**Tabla código**	**Tiempo promedio**	**Núm. parti.**	**Iter. espera**	**Aplica K-M**
A1	0.001167	4	4	3	A1	0.001015	4	3	3
A2	0.002656	2	3	3	A2	0.002550	3	2	0
A3	0.016059	15	10	3	A3	0.015084	15	9	3
A4	0.044856	8	10	3	A4	0.033120	9	5	3
A5	0.003292	6	8	3	A5	0.003212	7	5	5
A6	0.036781	10	10	3	A6	0.038804	12	10	5
A7	0.086573	30	35	3	A7	0.106526	45	25	3
A8	0.018262	4	4	3	A8	0.025914	6	5	3
A9	0.001499	5	5	3	A9	0.001424	5	5	3
A10	0.002909	2	3	3	A10	0.002649	2	2	2
A11	0.040849	20	25	3	A11	0.020999	20	10	3
A12	0.072730	12	10	3	A12	0.059045	12	8	3
A13	0.002305	5	7	3	A13	0.001702	5	4	4
A14	0.004915	3	5	3	A14	0.004158	3	3	2
A15	0.020600	15	10	3	A15	0.029647	17	17	4
A16	0.147504	16	15	3	A16	0.170197	20	15	4

Tabla 3.4: Resultados obtenidos en las 16 tablas de referencia en los algoritmos de búsqueda tabú, en 500 inicializaciones al azar.

	BT por transferencias				BT mediante moviendo CG		
Tabla	**%**	**Tiempo**	**MaxIte**	**Tabla**	**%**	**Tiempo**	**MaxIte**
A1	100 %	0.001518	6	A1	100 %	0.000604	2
A2	100 %	0.004347	3	A2	100 %	0.001673	1
A3	100 %	0.012865	30	A3	100 %	0.008768	15
A4	100 %	0.057998	22	A4	100 %	0.033548	10
A5	100 %	0.006412	18	A5	100 %	0.004625	10
A6	100 %	0.099558	60	A6	100 %	0.113792	50
A7	100 %	0.173089	350	A7	100 %	0.138560	210
A8	100 %	0.031910	10	A8	100 %	0.021573	5
A9	100 %	0.001975	8	A9	100 %	0.001416	5
A10	100 %	0.007709	5	A10	100 %	0.002317	1
A11	100 %	0.116413	220	A11	100 %	0.098313	150
A12	100 %	1.066081	275	A12	100 %	0.438988	85
A13	100 %	0.000962	3	A13	100 %	0.000747	2
A14	100 %	0.009698	7	A14	100 %	0.004688	3
A15	100 %	0.077000	130	A15	100 %	0.044830	60
A16	100 %	1.112870	300	A16	100 %	0.411317	80

Tal y como se constata en la tabla 3.4, de manera análoga a como sucedió con las heurísticas presentadas en la sección anterior, las heurísticas de búsqueda tabú lograron el 100 % de rendimiento en todas las tablas de referencia.

3.1.3. Resumen del rendimiento en las tablas de referencia

Para efectos de comparación, en la tabla 3.5 se realiza un resumen de los tiempos de ejecución de las heurísticas implementadas. En total cinco, en virtud de que se realizaron dos versiones para la heurística de enjambre de partículas (una por cada juego de parámetros indicado en el capítulo 2.4) y dos versiones para la heurística de búsqueda tabú (generación de vecinos mediante transferencias y por movimiento de centroides).

Tabla 3.5: Rendimiento de los algoritmos en las tablas de referencia.

Tabla	AG	EP-1	EP-2	BT-T	BT-CG
A1	0.001683	0.001167	0.001015	0.001518	0.000604
A2	0.003879	0.002656	0.002550	0.004347	0.001673
A3	0.040858	0.016059	0.015084	0.012865	0.008768
A4	0.139791	0.044856	0.033120	0.057998	0.033548
A5	0.011994	0.003292	0.003212	0.006412	0.004625
A6	0.129875	0.036781	0.038804	0.099558	0.113792
A7	0.556964	0.086573	0.106526	0.173089	0.138560
A8	0.074325	0.018262	0.025914	0.031910	0.021573
A9	0.001873	0.001499	0.001424	0.001975	0.001416
A10	0.005745	0.002909	0.002649	0.007709	0.002317
A11	0.080337	0.040849	0.020999	0.116413	0.098313
A12	0.222646	0.072730	0.059045	1.066081	0.438988
A13	0.003535	0.002305	0.001702	0.000962	0.000747
A14	0.020404	0.004915	0.004158	0.009698	0.004688
A15	0.126575	0.020600	0.029647	0.077000	0.044830
A16	2.093986	0.147504	0.170197	1.112870	0.411317

Analizando la tabla[1] 3.5 se pueden notar hechos como:

- Tomando como criterio el tiempo de ejecución para garantizar[2] un porcentaje de atracción del 100 % en este bloque de tablas, en promedio las tablas que podrían considerarse más difíciles son A7, A12 y A16. Siendo particularmente la A16 la más difícil.

- En búsqueda tabú y particularmente en las tablas de referencia, la estrategia de generación de vecinos mediante la transferencia de un individuo de una clase a otra, generó resultados de menor calidad que la estrategia de movimiento de centros de gravedad. Particularmente pueden observarse los tiempos de ejecución mostrados en la tabla 3.5 y para las tablas A7, A12 y A16.

- De las heurísticas implementadas, en las tablas de referencia enjambre de partículas fue la que generó mejores tiempos de ejecución. Particularmente, la versión EP-1 (tomando $\alpha = 0.7$, $c_1 = 1.25$ y $w = 1.75$), es la que presentó mejor calidad en los resultados.

- Como era de suponer, las características asociadas a la cardinalidad y la desviación estándar, son las que inciden mayoritariamente en el hecho que una tabla pueda ser catalogada difícil o no. Se evidencia que los algoritmos logran el agrupamiento óptimo de una manera más eficiente y con mejores tiempos de ejecución, en las tablas

[1]Se considera la notación BT-T para indicar BT por transferencias, BT-CG para BT mediante movimiento de centros de gravedad, EP-1 para enjambre de partículas con la configuración de parámetros $\alpha = 0.7$, $c_1 = 1.25$ y $w = 1.75$, y EP-2 para enjambre de partículas con el juego de parámetros $\alpha = 0.3$, $c_1 = 3.75$ y $w = 4.25$. Esta misma notación se emplea en las tablas 3.10 y 3.11.

[2]En el contexto de la presentación de resultados se utilizará el término "garantizar" como indicador de "alta probabilidad". Al estar trabajando con heurísticas de optimización, que corresponden a algoritmos no determinísticos, nunca se puede garantizar con certeza ningún porcentaje de atracción. Por ende, se procura tener escenarios relativamente estables, pero nunca certeros para una eventual replicación. En el caso de las tablas de referencia se tiene un óptimo de referencia, porque proviene de un experimento controlado.

cuyo agrupamiento natural de los datos se caracteriza porque las K clases tienen la misma cardinalidad y la misma desviación estándar.

3.2. Resultados en tablas adicionales

En lo consecuente se presentarán los resultados obtenidos en las ochos tablas adicionales diseñadas para probar los algoritmos. Dichas tablas fueron descritas en la sección 1.6 y codificadas en la tabla 3.6 para facilitar su referencia.

Tabla 3.6: Codificación de las tablas adicionales de 1050 y 2100 individuos.

Código	**Características de la tabla**	**Inercia de referencia**
A17	$n = 1050$, $K = 7$, Card(=) DS(=)	5.24360409974491
A18	$n = 2100$, $K = 7$, Card(=) DS(=)	5.2846294725285
A19	$n = 1050$, $K = 7$, Card($\neq$) DS(=)	5.30949287444044
A20	$n = 2100$, $K = 7$, Card($\neq$) DS(=)	5.2464823598562
A21	$n = 1050$, $K = 7$, Card(=) DS($\neq$)	22.0918880463104
A22	$n = 2100$, $K = 7$, Card(=) DS($\neq$)	22.5695921001011
A23	$n = 1050$, $K = 7$, Card($\neq$) DS($\neq$)	14.9465826139119
A24	$n = 2100$, $K = 7$, Card($\neq$) DS($\neq$)	15.5730443777207

Se indica nuevamente que las heurísticas de búsqueda tabú fueron ejecutadas con el parámetro ReiniSolActualCada $= 2$. En el caso de algoritmo genético la probabilidad de mutar y de cruce fueron fijados en $p_m = 65\,\%$ y $p_c = 15\,\%$, respectivamente. Para enjambre de partículas EP-1 se utilizaron los parámetros $\alpha = 0.7$, $c_1 = 1.25$ y $w = 1.75$, y en enjambres de partículas EP-2 se utilizó el juego de parámetros $\alpha = 0.3$, $c_1 = 3.75$ y $w = 4.25$.

En la tabla 3.7 se muestran los resultados generados por la heurística de algoritmo genético en las tablas A17, ..., A24. De manera similar, en la tabla 3.8 se resume el

rendimiento de los algoritmos de enjambres de partículas EP-1 y EP-2, en el mismo bloque de tablas.

Tabla 3.7: Resultados obtenidos en las tablas A17, ..., A24 en el algortimo genético, en 500 inicializaciones al azar.

Algoritmos Genéticos (tablas adicionales)					
Tabla código	**% de atrac.**	**Tiempo promedio**	**Núm. Insta.**	**Iter. espera**	**Aplica K-M**
A17	100 %	0.801.023	26	30	3
A18	100 %	0.834.842	20	15	2
A19	100 %	0.787.208	30	17	2
A20	100 %	1.060.325	23	16	2
A21	98.2 %	4.700.682	60	30	3
A22	98.4 %	6.490.773	35	30	2
A23	100 %	0.674.856	15	15	2
A24	99.6 %	5.054.125	25	40	2

Tabla 3.8: Resultados obtenidos en las tablas A17, ..., A24 en los algoritmos de enjambres de partículas, en 500 inicializaciones al azar.

Enjambre de partículas 1 (adicionales)						**Enjambre de partículas 2 (adicionales)**					
Tabla código	**% de atrac.**	**Tiempo promedio**	**Núm. parti.**	**Iter. espera**	**Aplica K-M**	**Tabla código**	**% de atrac.**	**Tiempo promedio**	**Núm. parti.**	**Iter. espera**	**Aplica K-M**
A17	100 %	0.197.615	13	12	3	A17	100 %	0.261.918	16	12	2
A18	100 %	0.357.421	12	12	3	A18	100 %	0.331.565	13	10	3
A19	100 %	0.210.658	12	12	3	A19	100 %	0.196.663	13	10	3
A20	100 %	0.409.605	11	12	3	A20	100 %	0.345.311	11	10	3
A21	94.8 %	2.779.674	45	45	2	A21	93.6 %	5.663.378	25	70	1
A22	100 %	1.982.523	17	17	1	A22	100 %	1.954.541	18	18	3
A23	100 %	0.898.955	20	20	3	A23	100 %	0.851.160	18	21	3
A24	100 %	1.248.418	10	30	3	A24	100 %	1.673.519	15	20	3

Por su parte, en la tabla 3.9 se muestran los resultados referentes a la aplicación de las heurísticas de búsqueda tabú en las tablas A17, ..., A24.

Tabla 3.9: Resultados obtenidos en las tablas A17, . . . , A24 en los algoritmos de búsqueda tabú, en 500 inicializaciones al azar.

BT por transferencias				**BT mediante moviendo CG**			
Tabla	**%**	**Tiempo**	**MaxIte**	**Tabla**	**%**	**Tiempo**	**MaxIte**
A17	100 %	0.199853	35	A17	100 %	0.126095	15
A18	100 %	0.413595	35	A18	100 %	0.295188	15
A19	100 %	0.359197	50	A19	100 %	0.225552	20
A20	100 %	0.656087	50	A20	100 %	0.449508	20
A21	100 %	0.582237	65	A21	100 %	0.384543	30
A22	100 %	9.782807	500	A22	100 %	1.803521	65
A23	100 %	0.354956	35	A23	100 %	0.149604	10
A24	100 %	1.419406	65	A24	100 %	0.340515	10

Para efectos de comparación, en la tabla 3.10 se realiza un resumen de los tiempos de ejecución de las heurísticas implementadas y específicamente en las tablas adicionales.

Tabla 3.10: Resumen del rendimiento de los algoritmos en las tablas A17, . . . , A24.

Tabla	**AG**	**EP-1**	**EP-2**	**BT-T**	**BT-CG**
A17	0.801023(100 %)	0.197615(100 %)	0.261918(100 %)	0.199853(100 %)	0.126095(100 %)
A18	0.834842(100 %)	0.357421(100 %)	0.331565(100 %)	0.413595(100 %)	0.295188(100 %)
A19	0.787208(100 %)	0.210658(100 %)	0.196663(100 %)	0.359197(100 %)	0.225552(100 %)
A20	1.060325(100 %)	0.409605(100 %)	0.345311(100 %)	0.656087(100 %)	0.449508(100 %)
A21	4.700682(98.2 %)	2.779674(94.8 %)	5.663378(93.6 %)	0.582237(100 %)	0.384543(100 %)
A22	6.490773(98.4 %)	1.982523(100 %)	1.954541(100 %)	9.782807(100 %)	1.803521(100 %)
A23	0.674856(100 %)	0.898955(100 %)	0.851160(100 %)	0.354956(100 %)	0.149604(100 %)
A24	5.054125(99.6 %)	1.248418(100 %)	1.673519(100 %)	1.419406(100 %)	0.340515(100 %)

Realizando un análisis del rendimiento de las cuatro heurísticas en las tablas A17, . . ., A24, particularmente con fundamento en la tabla 3.10, se pueden notar los siguientes hechos:

- Nuevamente la heurística de BT por movimiento de centroides muestra tiempos menores que la heurística de BT por transferencias. Particularmente es muy notorio en las tablas A22 y A24.
- Es bastante claro que para las heurísticas implementadas, las tablas A21, A22 y A24 son laa que presentas una mayor dificultad para la determinación del agrupamiento óptimo de los datos.
- Analizando los tiempos de ejecución, la heurística de BT mediante movimiento de centros de gravedad es la que genera resultados de mejor calidad. Particularmente muy notorio en A21 y A24.

3.3. Resultados en tablas clásicas de la literatura

Posterior a los análisis ya señalados, se procedió a ejecutar los algoritmos con cuatro tablas clásicas de la literatura del Análisis de Datos, a saber: *French Scholar Notes, Amiard's Fishes, Thomas's Sociomatrix* y *Fisher's Iris*. Dichas tablas constan respectivamente de 9, 23, 24 y 150 individuos, caracterizados por 5, 16, 24 y 4 variables cuantitativas, respectivamente. En cada una de las tablas se desconoce el agrupamiento natural de los datos, y al igual que en Trejos et al. (2004), y con el objetivo de establecer una comparación, se estudiarán los agrupamientos en 2, 3, 4 y 5 clases para las tablas *French Scholar Notes* y *Fisher's Iris*. Además, se analizarán los agrupamientos en 3, 4 y 5 clases para las tablas *Amiard's Fishes* y *Thomas's Sociomatrix*.

Para cada una de las heurísticas, las ejecuciones se realizaron con los mismos parámetros usados anteriormente. En la tabla 3.11 se muestran los resultados obtenidos en dichas pruebas. Para cada heurística, cada tabla y la cantidad respectiva de clases en la que se construye el agrupamiento, se muestra el tiempo promedio y el porcentaje de atracción cal-

culado a partir de 500 ejecuciones del algoritmo. De manera similar a como se realizó con las demás tablas, se empleó una tolerancia de 0.001 para la inercia reportada por cada algoritmo en esas ejecuciones, esto es, se consideró $\mathsf{TolInercia} = 0.001$. Las últimas tres columnas de la tabla 3.11 muestran los resultados reportados sobre estas mismas tablas en Trejos et al. (2004) y para las heurísticas de interés. Al respecto, W* denota la mejor inercia indicada en dicho artículo y sobre la cual reportan los porcentajes de atracción.

A partir de la información mostrada en la tabla 3.11 es dificil establecer diferencias significativas en el rendimiento de las heurísticas implementadas en la presente investigación. En términos globales en todas las tablas, y para los valores de K estudiados, los cinco algoritmos lograron determinar los agrupamientos óptimos en tiempos muy bajos. La única posible excepción corresponde a la tabla de *Thomas's Sociomatrix* con $K = 5$, en la que se reportan tiempos promedios mayores a un segundo.

Con respecto a los resultados reportados en Trejos et al. (2004) y dado que el único criterio disponible para efectos de comparación corresponde a los porcentajes de atracción, se considera que globalmente los resultados obtenidos en la presente investigación son de mayor calidad que los reportados en dicho artículo. No obstante, se advierte que dicha comparación tiene rasgos muy generales, dado a factores que se consideran fundamentales en el diseño de algoritmos, tales como las características del equipo computacional, los tiempos de ejecución (que a su vez dependen del equipo empleado), e incluso, las ligeras diferencias en algunas inercias reportadas que influyen tanto en el tiempo de ejecución, como en los porcentajes de atracción (véase por ejemplo la tabla de *Thomas's Sociomatrix* con $K = 5$).

Tabla 3.11: Rendimiento de los algoritmos en tablas clásicas de la literatura.

K	Inercia de referencia	AG	EP-1	EP-2	BT-T	BT-CG
French Scholar Notes (9 individuos, 5 variables)						
2	28.1902778	0.001197(100 %)	0.000508(100 %)	0.000493(100 %)	0.000664(100 %)	0.000901(100 %)
3	16.8148148	0.003979(100 %)	0.001146(100 %)	0.001184(100 %)	0.000927(100 %)	0.001812(100 %)
4	10.4675926	0.010065(100 %)	0.004549(100 %)	0.003853(100 %)	0.002783(100 %)	0.004054(100 %)
5	4.88888889	0.002739(100 %)	0.001171(100 %)	0.001257(100 %)	0.001142(100 %)	0.005087(100 %)
Amiard's Fishes (23 individuos, 16 variables)						
2	69368.2743	0.011433(100 %)	0.004313(100 %)	0.002481(100 %)	0.963748(3.2 %)	0.017696(100 %)
3	32213.3817	0.005557(100 %)	0.001206(100 %)	0.001227(100 %)	0.007207(100 %)	0.007603(100 %)
4	18281.3887	0.014077(100 %)	0.002431(100 %)	0.002394(100 %)	0.006504(100 %)	0.010618(100 %)
5	14497.8105	0.084217(100 %)	0.022874(100 %)	0.030277(100 %)	0.064127(100 %)	0.086814(100 %)
Thomas' Sociomatrix (24 individuos, 24 variables)						
2	333.767482	0.037998(100 %)	0.006211(100 %)	0.005918(100 %)	0.007322(100 %)	0.007265(100 %)
3	271.832639	0.026108(100 %)	0.010068(100 %)	0.010272(100 %)	0.019933(100 %)	0.020184(100 %)
4	235.025694	0.199395(100 %)	0.076755(100 %)	0.080285(100 %)	0.262970(100 %)	0.377637(100 %)
5	202.58125	1.271749(98.2 %)	1.126313(99.6 %)	1.288106(99.4 %)	1.646268(100 %)	1.760430(100 %)
Fisher's Iris (150 individuos, 4 variables)						
2	0.99916903	0.000678(100 %)	0.000550(100 %)	0.000698(100 %)	0.000412(100 %)	0.000381(100 %)
3	0.52136424	0.003068(100 %)	0.003310(100 %)	0.004581(100 %)	0.003923(100 %)	0.001221(100 %)
4	0.37817193	0.014703(100 %)	0.008865(100 %)	0.009084(100 %)	0.006780(100 %)	0.007019(100 %)
5	0.31204311	0.067068(100 %)	0.103283(100 %)	0.088706(100 %)	0.078735(100 %)	0.044485(100 %)

	Trejos et al. (2006)			
	W^*	AG	EP	BT
	28.2	100 %	92 %	100 %
	16.8	95 %	57 %	100 %
	10.5	97 %	51 %	100 %
	4.9	100 %	29 %	100 %
	-	-	-	-
	32213	87 %	51 %	100 %
	18281	0 %	23 %	100 %
	14497	0 %	6 %	97 %
	-	-	-	-
	271.8	85 %	7 %	100 %
	235.0	24 %	7 %	100 %
	202.6	0 %	7 %	98 %
	0.999	100 %	76 %	100 %
	0.521	100 %	79 %	76 %
	0.378	80 %	55 %	60 %
	0.312	6 %	28 %	32 %

3.4. Jerarquización de las heurísticas

En la presente investigación es de particular interés establecer una jerarquización de los algoritmos heurísticos implementados, como función del rendimiento que mostraron en las diferentes tablas de datos. Tomando como base la información resumida en las tablas 3.5, 3.10 y 3.11, mostradas en las páginas 119, 123 y 126, respectivamente, así como el análisis realizado en las secciones anteriores, se establece el siguiente nivel de prioridad entre las heurísticas:

Primer nivel

El algoritmo de enjambre de partículas bajo la combinación de parámetros $\alpha = 0.7$, $c_1 = 1.25$ y $w = 1.75$ (EP-1) y el algoritmo de búsqueda tabú mediante movimiento de centros de gravedad.

Entre estos dos algoritmos se advierte que existió evidencia, particularmente en los resultados expuestos en la tabla 3.10, sobre comportamientos más estables en el algoritmo de búsqueda tabú mediante movimiento de centros de gravedad, específicamente al aumentar la cantidad de individuos por clasificar.

Segundo nivel

Salvo por el tiempo reportado para la A12 en la tabla 3.5 que resume los tiempos de ejecución en las tablas de referencia, el algoritmo de búsqueda tabú por transferencias generó mejores resultados que el algoritmo genético[3]. En particular, en las tablas adicionales mostró mayor estabilidad en los tiempos de ejecución, mostrando menor sensibilidad ante el aumento de la cantidad de individuos por clasificar.

[3]Caso similar se reporta en la A22 en la tabla 3.10, pero con la diferencia que los $6.49s$ que reporta el algoritmo genético obedece a un 98.4 % de porcentaje de atracción. Este tiempo aumenta mucho más que los $9.78s$ que reporta el algoritmo BT-T, si se quisiera equiparar el 100 % de atracción que reporta esta última heurística en esa misma tabla.

En resumen, de acuerdo con la evidencia, las heurísticas implementadas se pueden jerarquizar en el siguiente orden:

1. Búsqueda tabú mediante movimiento de centros de gravedad.
2. Enjambre de partículas(EP-1).
3. Búsqueda tabú por transferencias.
4. Algoritmo genético.

3.5. Diseño de la heurística híbrida

El término híbrido hace referencia a un animal o vegetal procreado a partir de dos individuos de especies distintas. En este mismo sentido, y para este estudio, se consideró que un algoritmo heurístico híbrido corresponde a un método heurístico que resulta de la combinación, parcial o total, de dos métodos de optimización combinatoria, donde al menos uno de ellos es heurístico.

Bajo esta concepción, todos los métodos implementados en la investigación son considerados algoritmos heurísticos híbridos, puesto que el algoritmo base del algoritmo genético, de enjambre de partículas y de búsqueda tabú, fueron combinados con el método clásico de nubes dinámicas o $k-$medias. Sin embargo, como parte de los objetivos de la investigación se propuso diseñar e implementar una heurística híbrida, en el sentido de una combinación parcial o total de dos métodos de búsqueda basados en heurísticas.

Para el diseño del algoritmo híbrido se tomó la decisión de combinar una heurística poblacional con una basada en vecindarios. La elección anterior obedeció a tres razones primordiales. La primera de ellas es que un método poblacional realiza una exploración

robusta del espacio de soluciones factibles, debido a la comunicación existente entre los individuos encargados de realizar la exploración[4].

La segunda razón consiste en que los métodos basados en vecindarios son, por lo general, más rápidos en la exploración del espacio de soluciones factibles, aún cuando existen periodos en los que dicha exploración presenta características muy aleatorias. Por ende, la combinación de los dos tipos de heurísticas buscó rescatar, en algún sentido, las virtudes de los métodos seleccionados para la hibridación, tratando de mitigar sus debilidades.

La tercera, y última razón, consistió en que las heurísticas que generaron mejores rendimientos fueron una heurística problacional (EP-1) y una heurística basada en vecindarios (BT-CG). En virtud de lo anterior se tomó la decisión de hibridar el algoritmo de enjambre de partículas con el método de búsqueda tabú mediante movimiento de centros de gravedad.

En lo procedente se presentan los detalles del diseño y los resultados obtenidos en el algoritmo híbrido.

3.5.1. Descripción de la heurística híbrida

Para el diseño del híbrido se tomó como base la implementación de la heurística de enjambre de partículas, cuya descripción se puede consultar en el algoritmo 2 en la página 45 del presente reporte. A cada partícula del enjambre se le adicionó un método que se encarga de realizar la búsqueda tabú, a partir de la posición actual de la partícula y que empieza con una lista tabú vacía.

[4]Sin embargo, esta ventaja podría terminar repercutiendo negativamente en la velocidad del método, particularmente en tablas con mayor números de individuos por clasificar.

El método de búsqueda tabú aplicado a cada partícula es análogo al descrito e implementado durante la investigación, a excepción de la mejora de reinicio(ReiniSolActualCada). Los algoritmos 9 y 10, presentados en las páginas 86 y 87, describen los pormenores del método de búsqueda tabú que sirvió como base para la hibridación. La decisión de no incorporar la mejora de reinicio obedece a que en ese caso el método destruiría toda la información previa, dejando únicamente la información de la mejor posición de la partícula. Esto invalidaría el proceso de búsqueda previa realizada por la optimización mediante enjambre de partículas. Por lo demás, el manejo de la lista tabú, la generación del vecino mediante movimiento de centros de gravedad, la escogencia del vecindario y el criterio de aspiración, se realizaron de manera análoga a como se hizo en el algoritmo de búsqueda tabú.

En la implementación se estudiaron tres posibilidades:

- **Posibilidad 1:** Cada cierto número de iteraciones, sin que se reporte una mejora en la mejor solución encontrada por el enjambre, se aplica el método de búsqueda tabú a cada una de las partículas.

  Una vez que el enjambre de partículas se ha detenido en un óptimo local, esto es, que transcurre una cierta cantidad de iteraciones en las que el enjambre no reporta ninguna solución de mejor calidad, se ejecuta búsqueda tabú. Este método se aplica por igual a todas las partículas que conforman el enjambre, generando movimientos en los agentes que no obedecen a ningún comportamiento social.

  En el algoritmo 11 se puede observar el pseudocódigo de este diseño. La implementación del método de búsqueda tabú se muestra a partir de la línea 9, en donde el bucle PARA se encarga de aplicar la búsqueda tabú a cada partícula del enjambre de manera secuencial. Los valores de los parámetros utilizados fueron los mismos determinados en el capítulo 2.4.

- **Posibilidad 2:** Cada cierto número de iteraciones, sin que se reporte una mejora en la mejor solución encontrada por el enjambre, se aplica búsqueda tabú únicamente a la partícula que representa la mejor solución determinada por el enjambre.

  Dado que aplicar la búsqueda tabú en todas las partículas del enjambre (posibilidad 1) resultó costoso en términos del tiempo de ejecución cuando se utilizó un número relativamente grande de partículas, se decidió probar la modificación de aplicar búsqueda tabú únicamente a la partícula que represente la mejor solución encontrada.

  En la línea 9 del algoritmo 12 se puede observar la implementación de dicha variante. De manera similar al algoritmo 11, la ejecución del método de búsqueda tabú se hace cada cierta cantidad de iteraciones sin que el enjambre reporte una mejora. Por su parte, en la línea 10 se puede notar que la ejecución se hace únicamente a la mejor partícula (partícula líder del enjambre), mitigando de esta manera el alto consumo de tiempo de ejecución que significa aplicarlo a todas las partículas del enjambre.

- **Posibilidad 3:** Una vez que el método de optimización por enjambre de partículas ha convergido, se ejecuta la búsqueda tabú únicamente a la partícula que representa la mejor solución encontrada por el enjambre.

  Con el objetivo de acelerar el algoritmo, se decidió aplicar la técnica de búsqueda tabú una única vez durante la ejecución del programa. Esto se logra utilizando el método de búsqueda tabú como una herramienta de afinamiento de la solución generada por el algoritmo de partículas. Una vez que las partículas han determinado una buena solución, la misma es iterada por medio de búsqueda tabú con el objetivo de determinar una mejor solución por medio de una técnica de vecindarios.

  En el algoritmo 13 se observa el pseudocódigo de esta variante. En la línea 13 de dicho algoritmo se puede visualizar la única vez en la que se ejecuta la búsqueda tabú, justo antes de salir del procedimiento general. Una vez finalizado el proceso

correspondiente, el programa devuelve la mejor solución encontrada.

Algoritmo 11 Caso 1: Algoritmo híbrido, optimización por enjambres de partículas y búsqueda tabú

Entrada: Parámetros de entradas α, c_1, c_2.
Salida: La mejor posición encontrada por el enjambre.

1: Construye aleatoriamente el enjambre $\mathcal{E} = (\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m)$.
2: Contador $\leftarrow 0$.
3: **mientras** ContadorIteraSinMejora $<$ MaxIteraSinMejora **hacer**
4: Evaluar el costo de cada partícula: $W(\mathcal{P}_k)$, para $k = 1, 2, 3, \ldots m$
5: **para** $k \leftarrow 1$ **hasta** m **hacer**
6: Actualizar la posición de la partícula $\mathcal{P}_k$. Ver líneas 6 al 12 del algoritmo 7
7: **fin para**
8: Contador $\leftarrow$ Contador $+ 1$.
9: **si** ContadorIteraSinMejora es divisible entre AplicarBTCada **entonces**
10: **para** $k \leftarrow 1$ **hasta** m **hacer**
11: Aplicar búsqueda tabú a la partícula $\mathcal{P}_k$
12: **fin para**
13: **fin si**
14: **si** ContadorIteraSinMejora es divisible entre AplicarKMediasCada **entonces**
15: Aplique $k-$medias a cada una de las partículas del enjambre.
16: **fin si**
17: **fin mientras**
18: **retornar** La mejor solución encontrada durante la búsqueda.

3.5.2. Aplicación de la heurística híbrida

Para estudiar el rendimiento de la heurística híbrida se decidió realizar una serie de pruebas con una única tablas de datos. La tabla seleccionada fue la A16 (`Tabla-525(7) Card(≠)DS(≠)`), que de acuerdo con el análisis hecho en secciones previas es una de las tablas de tamaño medio que más presentó dificultad a los diferentes algoritmos.

Los parámetros de entrada que se utilizaron en el programa se resumen en la tabla 3.12. Estos valores se mantuvieron invariantes en las cinco aplicaciones realizadas para estas pruebas y que se detallarán posteriormente.

Algoritmo 12 Caso 2: Algoritmo híbrido, optimización por enjambres de partículas y búsqueda tabú

Entrada: Parámetros de entradas α, c_1, c_2.
Salida: La mejor posición encontrada por el enjambre.

1: Construye aleatoriamente el enjambre $\mathcal{E} = (\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m)$.
2: Contador $\leftarrow 0$.
3: **mientras** ContadorIteraSinMejora $<$ MaxIteraSinMejora **hacer**
4: Evaluar el costo de cada partícula: $W(\mathcal{P}_k)$, para $k = 1, 2, 3, \ldots m$
5: **para** $k \leftarrow 1$ **hasta** m **hacer**
6: Actualizar la posición del la k ésima partícula. Ver líneas 6 al 12 del algoritmo 7
7: **fin para**
8: Contador $\leftarrow$ Contador $+ 1$.
9: **si** ContadorIteraSinMejora es divisible entre AplicarBTCada **entonces**
10: Aplica búsqueda tabú a la partícula que representa la mejor solución determinada por el enjambre.
11: **fin si**
12: **si** ContadorIteraSinMejora es divisible entre AplicarKMediasCada **entonces**
13: Aplique $k-$medias a cada una de las partículas del enjambre.
14: **fin si**
15: **fin mientras**
16: **retornar** La mejor solución encontrada durante la búsqueda.

Algoritmo 13 Algoritmo de optimización por enjambres de partículas

Entrada: Parámetros de entradas α, c_1, c_2.
Salida: La mejor posición encontrada por el enjambre.

1: Construye aleatoriamente el enjambre $\mathcal{E} = (\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m)$.
2: Contador $\leftarrow 0$.
3: **mientras** ContadorIteraSinMejora $<$ MaxIteraSinMejora **hacer**
4: Evaluar el costo de cada partícula: $W(\mathcal{P}_k)$, para $k = 1, 2, 3, \ldots m$
5: **para** $k \leftarrow 1$ **hasta** m **hacer**
6: Actualizar la posición del la k ésima partícula. Ver líneas 6 al 12 del algoritmo 7
7: **fin para**
8: Contador $\leftarrow$ Contador $+ 1$.
9: **si** ContadorIteraSinMejora es divisible entre AplicarKMediasCada **entonces**
10: Aplique $k-$medias a cada una de las partículas del enjambre.
11: **fin si**
12: **fin mientras**
13: Aplica búsqueda tabú a la partícula que representa la mejor solución determinada por el enjambre.
14: **retornar** La mejor solución encontrada durante la búsqueda.

Tabla 3.12: Parámetros en la prueba de la heurística híbrida.

Parámetros de entrada	**Valor utilizado**
Cantidad de partículas en el enjambre	7
Iteraciones sin mejora para EP	10
Tamaño del vecindario	10
Tamaño de la lista tabú	5
Iteraciones sin mejora para BT	75
Número de corridas múltiples	100

Los parámetros correspondientes a la optimización por enjambre de partículas fueron tomados del ajuste previo realizado en el capítulo 2.4 y en particular referente a la heurística codificada con EP-1, caracterizada por el juego de parámetros $\alpha = 0.3$, $c_1 = 3.75$ y $W = 4.25$.

Para realizar una comparación objetiva, se vislumbraron cinco casos a estudiar, los cuales se describen a continuación:

Caso 1: Aplicación del método de búsqueda tabú a todas las partículas, esto cada **tres** iteraciones sin que se reporte un mejoramiento en la mejor solución encontrada por el enjambre. La mejora de $k-$medias permanece inactiva.

Caso 2: Aplicación del método de búsqueda tabú a la partícula que representa la mejor solución encontrada por todo el enjambre. La aplicación de esta mejora se hizo cada **tres** iteraciones sin que se reportara un mejoramiento en la mejor solución encontrada por el enjambre. La mejora de $k-$medias permanece inactiva.

Caso 3: Aplicar la mejora de búsqueda tabú a la partícula que represente la mejor solución encontrada por el enjambre. Se realiza al finalizar la exploración de las partículas

y se ejecuta una única vez en cada corrida del programa. La mejora de $k-$medias permanece inactiva.

Caso 4: No aplicar la mejora de búsqueda tabú, pero activando la mejora de nubes dinámicas o $k-$medias cada **tres** iteraciones sin que se reporte una mejora en la mejor solución encontrada por el enjambre.

Caso 5: Aplicar la optimización por enjambres de partículas, sin la aplicación del método de búsqueda tabú, ni la mejora de $k-$medias.

Los porcentajes de atracción y tiempos promedios de ejecución para cada uno de los casos anteriores se resumen en la tabla 3.13.

Tabla 3.13: Resultados obtenidos en los casos en análisis.

	Caso 1	**Caso 2**	**Caso 3**	**Caso 4**	**Caso 5**
% de atracción	99 %	95 %	89 %	79 %	23 %
Tiempo promedio	16.971.294s	1.558.660s	0.432.240s	0.022.208s	0.022.207s

3.5.3. Análisis del rendimiento

De la tabla 3.13 se pueden realizar varias observaciones que se procederán a discutir a continuación.

La implementación de la búsqueda tabú al algoritmo de enjambre de partículas aumentó considerablemente el porcentaje de atracción. Tomando como base únicamente este criterio, de las tres propuestas sobre la incorporación de búsqueda tabú al algoritmo de enjambre de partículas, aplicarla a todas las partículas cada tres iteraciones sin mejora (Caso 1), es por mucho la mejor alternativa, dado que se reporta un 99 % de porcentaje de atracción. El segundo lugar lo constituye la estrategia explicada en el Caso 2, con un

95 %, mientras que la estrategia concerniente al Caso 3 generó un 89 % de atracción. En cualquier caso, las tres aplicaciones mejoran por mucho el 23 % de atracción que generó el algoritmo de enjambres sin búsqueda tabú. Incluso mejoran el porcentaje de atracción generado con la aplicación de $k-$medias en el algoritmo de enjambres, la cual obtiene un 79 %.

Desde el punto de vista de porcentajes de atracción, se puede considerar que el algoritmo híbrido generó resultados satisfactorios en sus tres alternativas de aplicación. Sin embargo, es posible notar un aumento significativo en el tiempo promedio de ejecución. Para la alternativa expuesta en el primer caso, y que generó el mejor porcentaje de atracción, se reporta que en promedio cada ejecución tarda $16.97s$. El segundo y tercer caso obtienen tiempos de $1.56s$ y $0.43s$, respectivamente. Sin embargo, el no aplicar ninguna mejora o bien, aplicar $k-$medias cada tres iteraciones sin mejora, generó tiempos promedios similares y que rondan los $0.022s$. De esto último se concluye que aplicar la mejora de $k-$medias no tiene mayor incidencia en el tiempo promedio de ejecución, pero sí tiene un incidencia positiva en el aumento del porcentaje de atracción.

Dada esa incidencia positiva que muestra la mejora de $k-$medias (poco impacto en el tiempo de ejecución), es que se decidió para tener un insumo adicional de comparación y poder medir la pertinencia global de la aplicación de búsqueda tabú al algoritmo de enjambres, analizar el porcentaje de atracción generado por la mejora de $k-$medias, pero aumentado el número de partículas. Este incremento en la población de partículas aumentó tanto el tiempo promedio de ejecución, como el porcentaje de atracción.

En la ejecución se mantuvieron los mismos parámetros para el enjambre, la mejora de $k-$medias cada tres iteraciones sin mejora y se mantuvo desactivado el complemento asociado a búsqueda tabú (para poder establecer una comparación posterior). Los resultados de esta nueva ejecución del algoritmo se pueden observar en la tabla 3.14.

Tabla 3.14: Resultados obtenidos con EP, aplicando $k-$medias, desactivando búsqueda tabú e incrementando el número de partículas en el enjambre.

Resultados	**Valor reportado**
Número de partículas	17
% de atracción	99 %
Tiempo promedio	0.099.216s

En los resultados expuestos en la tabla 3.14, se puede observar que dicha ejecución iguala o mejora los resultados obtenidos al complementar el algoritmo de enjambre de partículas con el algoritmo de búsqueda tabú. A pesar de que el porcentaje de atracción es el mismo, el tiempo de ejecución es mucho menor. Por esta razón se concluye que el algoritmo híbrido generó buenos resultados, pero globalmente su rendimiento no fue significativo, dado que es más rentable aumentar el número de partículas, obteniendo con ello resultados similares al algoritmo con búsqueda tabú, pero en menores tiempos de ejecución.

Capítulo 4

Apartado 11: Discusión y conclusiones

En el presente capítulo se resumen los principales hechos que surgieron como consecuencia del estudio. En primera instancia se discute sobre el alcance de los objetivos propuestos para la investigación.

4.1. Alcance de los objetivos

En función de la información claramente detallada en este informe, se concluye que en términos generales los objetivos propuestos fueron alcanzados satisfactoriamente. Como evidencias se sintetizan las siguientes:

1. Se logró la implementación de tres heurísticas de optimización combinatoria para estudiar el problema de particionamiento de individuos en presencia de datos cuantitativos (algoritmo genético, enjambre de partículas y búsqueda tabú). Con

la característica adicional que la heurística de búsqueda tabú fue implementada siguiendo dos estrategias diferentes para la generación de vecinos. Esto potenció el haber culminado con cuatro algoritmos heurísticos para el estudio del problema en cuestión (PRIMER OBJETIVO ESPECÍFICO).

2. Del proceso de investigación se derivaron un conjunto de estrategias que potencializaron el rendimiento de las heurísticas. Como ejemplos particulares se pueden citar: la mejora de $k-$medias en todas las heurísticas, el método de paro en todas las heurísticas, la mejora de reinicio en las heurísticas de búsqueda tabú y la definición del operador de cruce y el operador de selección en el algoritmo genético.

3. Se realizó un análisis de los parámetros en cada uno de los algoritmos, del cual se desprendieron los valores que deben asignarse a los diferentes parámetros de las heurísticas para potenciar su rendimiento (SEGUNDO OBJETIVO ESPECÍFICO).

4. Se realizó una comparación del rendimiento, con respecto a los porcentajes de atracción y tiempos de ejecución, de las tres heurísticas implementadas en la investigación. Esto permitió jerarquizar a las heurísticas de búsqueda tabú mediante movimiento de centros gravedad (BT-CG) y enjambre de partículas en su versión EP-1, como las heurísticas de mejor rendimiento. Aportando evidencia que al aumentar significativamente la cantidad de individuos que conforman la tabla de datos, es la heurística BT-CG la que experimenta menos aumento en los tiempos de ejecución (TERCER OBJETIVO ESPECÍFICO).

5. Se diseñó una heurística híbrida entre las dos mejores heurísticas, a saber entre enjambre de partículas y búsqueda tabú (CUARTO OBJETIVO ESPECÍFICO). Se advierte que a pesar de las diferentes aristas exploradas y que el algoritmo híbrido generó buenos porcentajes de atracción, no cumplió con la expectativa de mejorar las heurísticas previamente implementadas. Los altos tiempos de ejecución fue un

punto que en cierta medida hace concluir que el proceso de hibridación no fue del todo exitoso.

4.2. Otras conclusiones

En esta sección se resumen otras conclusiones desprendidas del proceso de investigación. Algunas de ellas fueron discutidas en la sección anterior.

- La aplicación del algoritmo de k-medias, en su plenitud o parcialmente (dependiendo de la heurística), complementó de una manera muy efectiva a las heurísticas diseñadas. Dicho algoritmo ayudó a mejorar considerablemente los tiempos de ejecución y los porcentajes de atracción.
- En los algoritmos de búsqueda tabú, la estrategia de reiniciar periódicamente la solución vigente en una determinada iteración, desde puntos aleatorios del espacio de búsqueda, mejoró considerablemente el rendimiento de dichas heurísticas.
- En las heurísticas de búsqueda tabú implementadas en la presente investigación se concluyó que el valor óptimo para el parámetro ReiniSolActualCada es 2.
- Es necesario profundizar sobre el diseño de una estrategia de intensificación en los algoritmos de búsqueda tabú. A pesar de que se obtuvieron buenos resultados al aplicar dichas heurísticas a las diferentes tablas de datos, se tiene la evidencia bibliográfica sobre la pertinencia de ejecutar un proceso de intensificación. Por ello se manifiesta la necesidad de estudiar este fenómeno en procesos futuros de investigación.
- En los algoritmos de búsqueda tabú, la estrategia de generación de vecinos mediante movimiento de centros de gravedad, generó mejores resultados que la estrategia de transferencias de un individuo de una clase a otra.

- El método de paro empleado en las heurísticas de algoritmo genético y enjambre de partículas de "esperar un cierto número de iteraciones sin que se reporte una mejora en la mejor clasificación encontrada por el método", mostró generar buenos resultados en dichas heurísticas. Este método de paro parece evitar la convergencia prematura al dar tiempo para que los algoritmos puedan escapar de óptimos locales.

- Para el algoritmo genético, los operadores utilizados de cruce (mejor padre hereda la clase más populosa) y de selección (SUS), generaron buenos resultados. Sin embargo, se recomienda realizar una comparación con métodos de cruce y selección clásicos para poder establecer su efectividad.

- Los parámetros de mutación y de cruce para el algoritmo genético que generaron buenos resultados en la investigación fueron, respectivamente, $p_m = 0.65$ y $p_c = 0.15$. Estos parámetros, si bien no son los recomendados por la literatura, demostraron su eficiencia al incrementar el rendimiento del algoritmo genético considerablemente para los experimentos realizados.

- Los parámetros w y c_1 que generaron los mejores resultados en el algoritmo de enjambre de partículas, en las tablas utilizadas para la optimización de parámetros, cumplen la relación

$$w = c_1 + 0.5,$$

  de donde se concluye que el parámetro social c_2 debe tener un valor de 0.5. El parámetro c_1 puede ser tomado en el intervalo $[0, 4.5]$. Por otro lado, el parámetro de tendencia α se debe tomar positivo y cercano a cero. Si bien se recomienda ajustar α para diferentes escogencias de los parámetros c_1 y w, este parámetro puede ser escogido en el conjunto $]0, 1]$. Lo anterior, pues el rendimiento de la heurística decayó considerablemente para valores de α superiores a la unidad.

- Los dos juegos de parámetros escogidos y que generaron buenos resultados durante

la ejecución de la heurística de enjambre de partículas son los que se presentan en la tabla 4.1.

Tabla 4.1: Combinación de parámetros generados en el algoritmo de enjambre de partículas.

Primer conjunto de parámetros		Segundo conjunto de parámetros	
Parámetro	**Valor propuesto**	**Parámetro**	**Valor propuesto**
w	1.75	w	4.25
c_1	1.25	c_1	3.75
α	0.7	α	0.3

En las pruebas realizadas se generaron mejores resultados utilizando el primer conjunto de parámetros que el segundo.

- En la investigación surgió evidencia que apoya que el número de variables que describen a los individuos tiene mayor incidencia en el rendimiento de las heurísticas que el número de individuos por clasificar. Esto es, a mayor cantidad de variables mayor dificultad para los algoritmos. Además, para los algoritmos es mucho más simple determinar el agrupamiento cuando el valor de K usado responde al agrupamiento natural de los datos, en contraposición cuando se trata de determinar agrupamientos no naturales.

- Las características asociadas a la cardinalidad y la desviación estándar, son las que inciden mayoritariamente en el hecho que una tabla pueda ser catalogada difícil o no. Se evidencia que los algoritmos logran el agrupamiento óptimo de una manera más eficiente y con mejores tiempos de ejecución, en las tablas cuyo agrupamiento natural de los datos se caracteriza porque las K clases tienen la misma cardinalidad y la misma desviación estándar. Siendo particulamente la desviación estándar usada a lo interno de las clases, el factor que influye mayoritariamente en el rendimiento de las heurísticas.

Es importante resaltar que todas las heurísticas diseñadas generaron resultados de mucha calidad en las diferentes tablas de datos en las que fueron ejecutadas. Sin embargo, dada la necesidad de establecer una priorización, entonces en función de los resultados presentados en el capítulo 3, se determina que las heurísticas que mostraron mejores resultados globales corresponden a búsqueda tabú mediante movimiento de centros de gravedad y enjambre de partículas (con la combinación de parámetros identificada con el código EP-1).

En resumen, las heurísticas implementadas fueron jerarquizadas (ver sección 3.4) en el siguiente orden:

1. Búsqueda tabú mediante movimiento de centros de gravedad.
2. Enjambre de partículas(EP-1).
3. Búsqueda tabú por transferencias.
4. Algoritmo genético.

Dado que la heurística EP-1 generó mejores tiempos en las tablas de referencia (tablas de 105 y 525 individuos), en contraposición al hecho que BT-CG generó mejores tiempos de ejecución en la tablas adicionales (de 1050 y 2100 individuos), esto genera evidencia a favor de que las heurísticas basadas en multiagentes experimentan aumentos significativos en los tiempos de ejecución al aumentar la cantidad de individuos que conforman la tabla de datos. Parece que este efecto es significativamente menor en las heurísticas de vecindario, que en el caso de la investigación se evidenció a partir de la heurística de búsqueda tabú mediante movimiento de centros de gravedad.

Capítulo 5

Apartado 12: Recomendaciones

En futuros procesos de investigación podrían abordarse otras heurísticas de optimización que no fueron contempladas en el presente proyecto, por ejemplo, puede considerarse opciones como los algoritmos de sobrecalentamiento simulado (Simulated Annealing) y colonias de hormigas (Ant Colony Optimization).

Por otra parte, dado que en el diseño del algoritmo híbrido se tuvo el principal inconveniente del aumento significativo de los tiempos de ejecución, podría explorarse la posibilidad de hibridación de la heurística de búsqueda tabú, con alguna otra heurística también basada en vecindarios, en particular sobrecalentamiento simulado podría ser una buena opción. En particular, la característica de las heurísticas de vecindarios de iterar cada vez una única solución factible, mostró en búsqueda tabú una menor sensiblidad, en términos de los tiempos de ejecución, ante el incremento de la cantidad de individuos que conforman las tablas de datos.

Finalmente, en futuras investigaciones con esta misma línea, es oportuno coordinar con profesionales de otras escuelas del Instituto Tecnológico, para considerar la posibilidad de contar con tablas de datos relacionadas al quehacer de los investigadores del TEC, y

que respondan idealmente a fenómenos más aplicados a la realidad. De esta manera, se buscaría contextualizar el Análisis de Datos, y en particular, el particionamiento de datos cuantitativos, a conjuntos de datos más actualizados y que respondan a un contexto.

Bibliografía

Abbass, H.; Sarker, R. & Newton, C. (2002). *Data mining: A heuritic approach*. Idea Group Publishing, Hershey.

Alias, M.; Aziz, N.; Aziz, K. & Mohemmed, A. (2011). "Particle swarm optimization for constrained and multiobjective problems: a brief review", *International Conference on Management and Artificial Intelligence* **6**: 146–150.

Babu, P. & Murty, N. (1994). "Simulated annealing for selecting optimal initial seeds in the k-means algorithm", *Indian Journal Pure and Applied Mathematics* **25**(1 y 2): 85–94.

Bagirov, A. & Mardeneh, K. (2006). "Modified global k-means algorithm for clustering in gene expression data sets", en: M. Bodén & T. Bailey (Eds.), *Conferences in Research and Practice in Information Technology (CRPIT)*, Australian Computer Society, Inc., Australia.

Berzal, F. (2005). "Métodos de agrupamiento: clustering", en: http://elvex.ugr.es/doc/proyecto/cap8.pdf, consultado el 28/06/2013, 11:38 a.m.

Bratton, D. & Kennedy, J. (2007). "Defining a standard for particle swarm optimization", en: IEEE, editor, *IEES Swarm Intelligence Symposium*, Institute of Electrical and Electronics Engineers, Hawaii: 120–127.

Brusco, M. (1999). "Orph-based local-search heuristics for large-scale combinatorial data analysis", *Journal of Classification* **16**: 163–180.

Clerc, M. (1999). "The swarm and the queen: Towards a determininistic and adaptive particle swarm optimization", en *Congress on Evolutionary Computation*, CEC99: 1951–1957.

de los Cobos, S.; Goddard, J.; Gutiérrez, M. & Martínez, A. (2010). *Búsqueda y exploración estocástica*. México Editorial CBI, México D.F.

Eberhart, R. & Kennedy, J. (1995). "A new optimizer using particle swarm theory", en *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, IEEE service center, Piscataway, NJ, Nagoya, Japan: 39–43.

Gendreau, M. & Potvin, J. (2008). "Tabu search", en: http://www.inf.ufpr.br/aurora/disciplinas/topicosia2/livros/search/TS.pdf, consultado el 17/03/2013, 03:13 p.m.

Gestal, M. (s.f.). "Introducción a los algoritmos genéticos", en: http://sabia.tic.udc.es/mgestal/cv/aaggtutorial/tutorialalgoritmosgeneticos.pdf, consultado el 18/07/2013, 09:19 p.m.

Gestal, M.; Rivero, D.; Rabuñal, J.; Dorado, J. & Pazos, A. (2010). *Introducción a los algoritmos genéticos y la programación genética*. Universidade da Coruña, Servizo de Publicacións, A Coruña.

Gil, N. (2006). "Algoritmos genéticos", en: http://www.monografias.com/trabajos-pdf/algoritmos-geneticos/algoritmos-geneticos.pdf, consultado el 17/07/2013: 09:24 p.m.

Glover, F. (1989). "Tabu search−part i", *Journal on Computing* **1**(3): 190–206.

Glover, F. (1990). "Tabu search−part ii", *Journal on Computing* **2**(1): 4–32.

Griffihs, A.; Miller, J.; Suzuki, D.; Lewontin, R. & Gelbart, W. (2000). *Genética*. W. H. Freeman, New York.

Hassan, R.; Cohanim, B. & Weck, O. (2005). "A comparison of particle swarm optimization and the genetic algorithm", en: G. Venter, editor, *Proceedings of the First AIAA Multidisciplinary Design Optimization Specialist Conference*, American Institute of Aeronautics and Astronautics, Austin, TX: 1–13.

Hvass, M. & Chipperfield, A. (2010). "Simplifying particle swarm optimization", *Applied Soft Computing* **10**(2): 618–628.

Kennedy, J. & Eberhart, R. (2001). *Swarm intelligence*. Academic Press, San Diego, CA.

Lee, K. & El-Sharkawi, M. (2008). *Modern heuristic optimization techniques*. John Wiley & Sons, Hoboken, NJ.

Lima, J. & Barán, B. (2006). "Optimización de enjambre de partículas aplicada al problema del cajero viajante bi-objetivo", *Revista Iberoamericana de Inteligencia Artificial* **10**(32): 67–76.

Mitchell, M. (1999). *An introduction to genetic algorithms*. MIT Press, Cambridge, MA.

Moujahid, A.; Inza, I. & Larrañaga, P. (s.f.). "Algoritmos genéticos", en: http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t2geneticos.pdf, consultado el 26/06/2013, 10:12 p.m.

Murillo, A. (2000). "Aplicación de la búsqueda tabú en la clasificación por particiones", *Revista Ciencias Matemáticas* **21**(3): 183–194.

Murillo, A. & Trejos, J. : 1–26.

Ng, M. & Wong, J. (2002). "Clustering categorical data sets using tabu search techniques", *Elsevier Science Ltd on behalf of Pattern Recognition Society* **35**(12): 2783–2790.

Peinado, J.; Iglesias, C. & Frailes, M. (2003). "Arquitectura de un sistema de optimización accesible a través de servicios web XML", en: http://eprints.ucm.es/8726/1/Memoria.pdf, consultado el 19/07/2013, 10:47 a.m.

Piza, E.; Murillo, A. & Trejos, J. (1999). "Nuevas técnicas de particionamiento en clasificación automática", *Revista de Matemática: Teoría y aplicaciones* **6**(1): 51–66.

Riojas, A. (2005). "Conceptos, algoritmo y aplicación al problema de las n−reinas", Tesis de Licenciatura, Universidad Nacional de San Marcos, Lima.

Rudolph, G. (1994). "Convergence analysis of canonical genetic algorithms", *IEEE Transactions Neural Netwoks* **5**(1): 96–101.

Sarkar, M. & Yegnanarayana, B. (1996). "A clustering algorithm using evolutionary programming", en *Proc. Int. Conf. Neural Networks, ICNN-96*, IEEE International Conference on Neural Networks, Washington, DC: 1162–1167.

Sedighizadeh, D. & Masehian, E. (2009). "Particle swarm optimization methods, taxonomy and applications", *International Journal of Computer Theory and Engineering* **1**(5): 486–502.

Settles, M. (2005). "An introduction to particle swarm optimization", en: http://www2.cs.uidaho.edu/~tsoule/cs504/particleswarm.pdf, consultado el 01/06/2013, 01:25 p.m.

Talbi, E. (2009). *Metaheuristics: From design to implementation.* John Wiley & Sons, New Jersey.

Trejos, J.; Castillo, W. & González, J. (2014). *Análisis multivariado de datos: Métodos y aplicaciones.* Editorial de la Universidad de Costa Rica, San José, primera edition.

Trejos, J. & Murillo, A. (2004). "Heuristics of combinatorial optimization and applications to data analysis", en *Proceeding of the I Summer School on Optimization and Numerical Analysis*, Institut für Mathematik, Berlin.

Trejos, J.; Murillo, A. & Piza, E. (1998). "Global stochastic optimization techniques applied to partitioning", en: A. Rizzi; M. Vichi & H. Bock (Eds.), *Advances in Data Science and Classification*, Springer, Berlin: 185–190.

Trejos, J.; Murillo, A. & Piza, E. (2004). "Clustering by ant colony optimization", en: D. Banks; F. McMorris; P. Arabie & W. Gaul (Eds.), *Classification, Clustering, and Data Mining Applications*, Springer Berlin Heidelberg, Berlin: 25–32.

Trejos, J.; Piza, E.; Murillo, A. & Pacheco, A. (2006). "Comparison of metaheuristics for partitioning in cluster analysis", *Investigación Operacional* **27**(2): 124–128.

Trejos, J. & Villalobos, M. (2007). "Partitioning by particle swarm optimization", en: P. Brito; G. Cucumel; P. Bertrand & F. Carvalho (Eds.), *Selected Contributions in Data Analysis and Classification*, Springer, Berlin: 235–244.

Umarani, R. & Selvi, V. (2010). "Particle swarm optimization-evolution, overview and applications", *International Journal of Engineering Science and Technology* **2**(7): 2802–2806.

Valvert, J. (2006). "Métodos y técnicas de reconocimiento de rostros en imágenes digitales bidimensionales", en: http://biblioteca.usac.edu.gt/tesis/08/08_0310_CS.pdf, consultado el 01/06/2013, 03:05 p.m.

Winker, P. (2001). *Optimization heuristics in econometrics: Applications of threshold accepting*. John Wiley & Sons, New York.

Xie, X.; Zhang, W. & Yang, Z. (2002a). "Adaptive particle swarm optimizationon individual level", en *Signal Processing, 2002 6th International Conference on*, International Conference on Signal Processing (ICSP), Beijing, China: 1215–1218.

Xie, X.; Zhang, W. & Yang, Z. (2002b). "A dissipative particle swarm optimization",

en *Proceedings of the IEEE Congress on Evolutionary Computation*, IEEE Computer Society Washington, DC, Honolulu: 1456–1461.